支払備金の見積もりにおける機械学習・深層学習的手法の適用と従来手法との比較（2018 ASTIN WP 参加報告）

アクサ生命保険株式会社　平松　雄司

【司会】　時間となりましたので、セッションＢ、「支払備金の見積もりにおける機械学習・深層学習的手法の適用と従来手法との比較（2018 ASTIN WP 参加報告）」を開始します。発表者は、アクサ生命の平松さんです。それでは、よろしくお願いします。

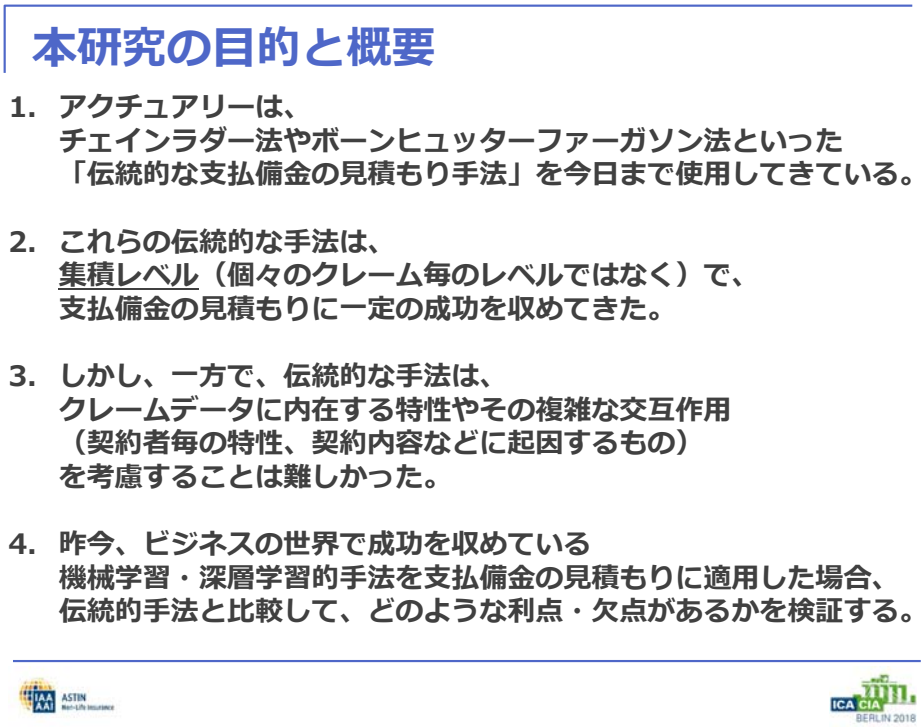
【平松】　よろしくお願いいたします。アクサ生命の平松と申します。本日は、お忙しい中お集まりいただき、ありがとうございます。本発表では、昨年末から今年の初めにかけまして参加いたしました、ASTIN 国際ワーキングパーティーの参加報告をさせていただきます。こちらが、本日のアジェンダとなります。

ASTIN WP の参加報告に加えまして、最後に、ASTIN 関連研究会の活動内容ですね。そちらの方も軽くさせていただきますが、１時間程おつきあいいただけましたらと思います。

最初に、ASTIN WP について、ご説明をさせていただきます。今年の６月に、４年に１度開催されます国際アクチュアリー会議が、ベルリンにて行われました。ASTIN WP は、開催期間中の月曜日の午後のセッションにおきまして、「Machine Learning and Traditional Methods Synergy in Non-Life Reserving」と題しまして、支払準備金の見積もりにおける機械学習と深層学習的手法の適用に関する検討内容を発表いたしました。この取り組みは、2014 年に発足し、教師あり・教師なし学習に関しまして検討いたしましたワ

ーキングパーティーや、2016 年に発足し、ニューラルネットワークを支払備金の見積もりに適用する、そのような検討をいたしましたワーキングパーティーに続くものとなっております。

ワーキングパーティーの方なのですが、メンバーは、各国から参加した総勢 34 名で構成されておりまして、日本からは、私を含めまして３人、ASTIN 関連研究会より参加いたしました。コミュニケーションツールは、skype や slack。また、プログラムコードは R。そして、そちらのプログラムコードの管理は、GitHub を利用いたしました。後ほど、こちらの各ツールに関しまして、簡単にご紹介をさせていただきます。また、これは少し雑感めいたもので恐縮なのですが、日本は、GMT＋９ですので、skype のミーティング時間が夜中の２時、３時ということで、非常に大変でした。



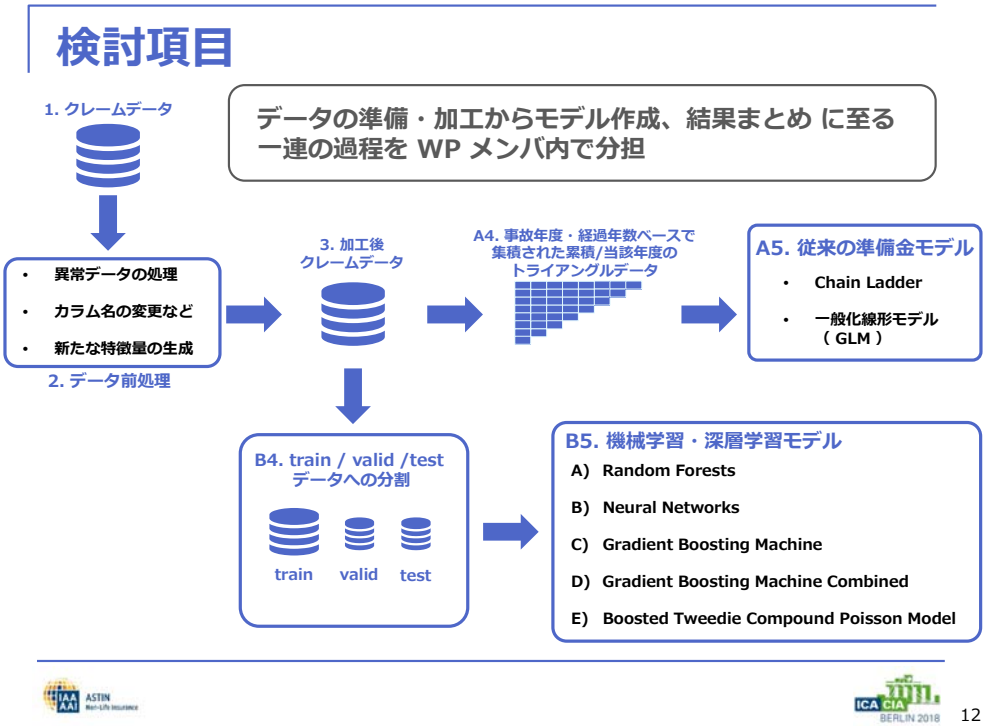
それでは、次に、本ワーキングパーティーの研究目的と概要に関しまして、ご説明をさせていただきます。アクチュアリーなのですけども、今日まで、支払備金の見積もり手法として、チェインラダー法やボーンヒュッターファーガソン法などの手法を伝統的に使用いたしております。これらの伝統的な手法は、ここでは集積レベルと申し上げておりますが、基本的にはトライアングルデータの形にしたものを指しております。そのような集積レベルでの支払備金の見積もりにおいて、一定の成功を収めてきていると言えると思います。

しかし、一方で、伝統的な手法は、個々のクレームデータに内在しておりますような特性や、複雑な交互作用といったものを考慮することは非常に難しいという欠点がございます。そこで、本ワーキングパーティーでは、他の業界、例えばＩＴ業界のウェブマーケティングなどの分野で成功を収めております機械学習・深層学習的手法を支払備金の見積もりに適用した場合、伝統的手法と比較して、どのような利点・欠点があるかを検証いたしました。

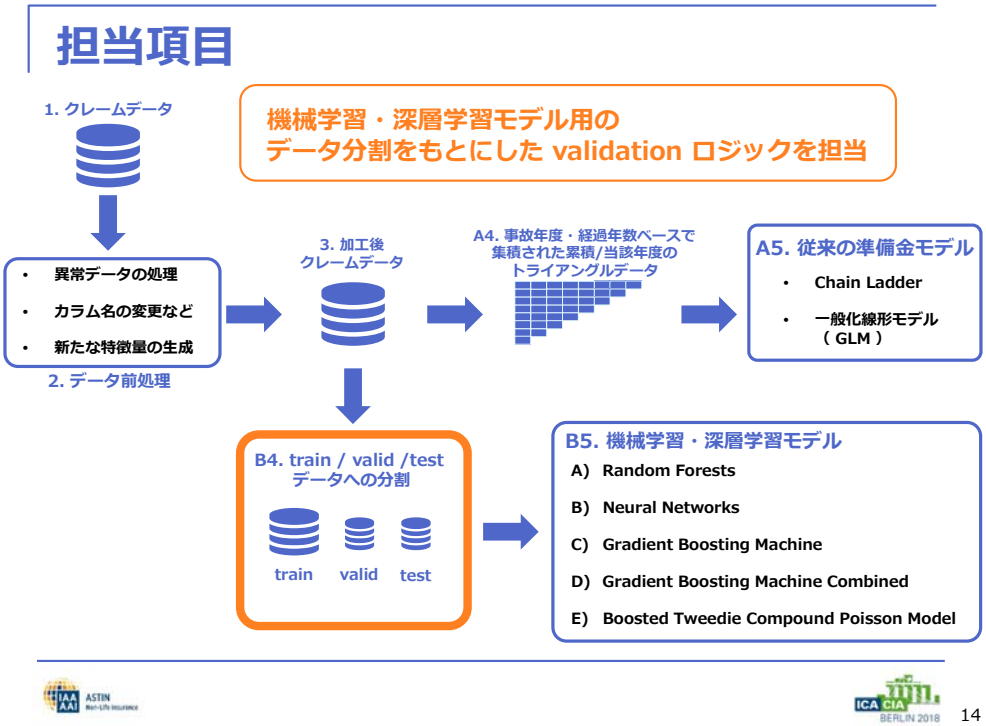
それでは、詳細な検討内容に入る前に、先ほど少しご紹介いたしましたが、本ワーキングパーティーで使用いたしました各種ツールのお話を、簡単にさせていただきます。まず、slack ですけれども、現在、日本でも多くのビジネスで使用されるようになってきましたコミュニケーションツールです。主に IT 関連の会社等で、社内プロジェクトで頻繁に使用されているとは思いますけれども、一部の保険会社の皆様方のもとでも使われているものかと思います。本ワーキングパーティーも、slack において、メンバー間のやり取りなどを、意思疎通を行っておりました。

また、R のプログラムコードですけれども、コードの一意性ですね。つまり、何が最新なのか、バージョ

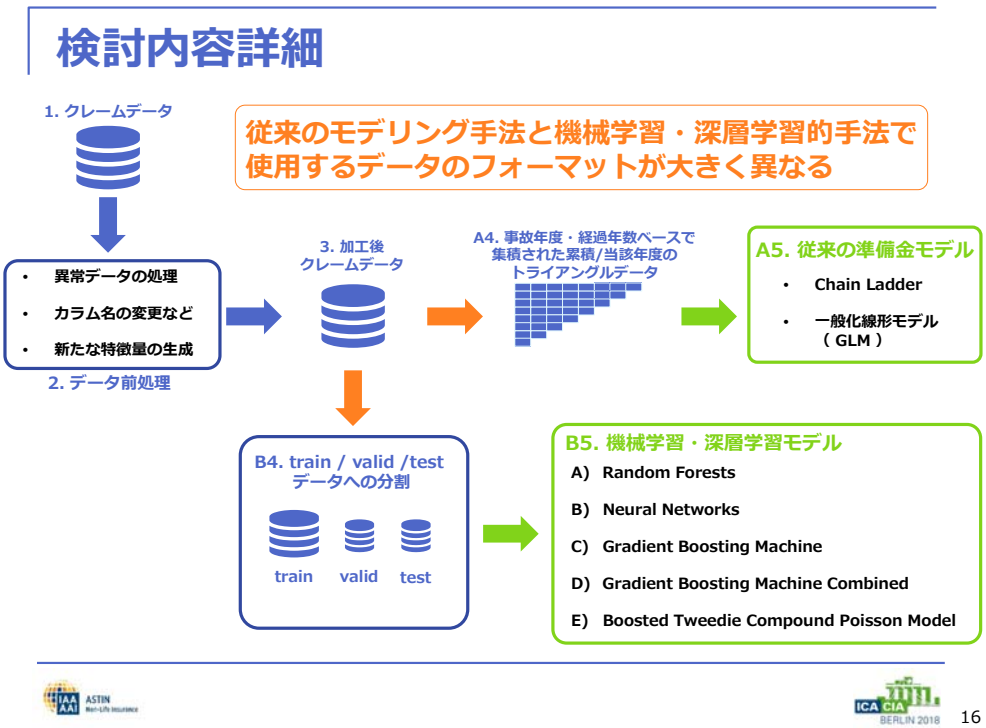
ンが古いコードは何なのか。そのようなところをしっかり管理するために、GitHub でコード管理をいたしておりました。残念ながら、こちらは現在もプライベート管理状態ですので、一般公開はしていないということで、そちらをご承知いただけましたらと思います。



では、本発表の本題であります、検討内容の説明に入っていきたいと思います。本ワーキングパーティーのアウトプットとして目指しておりましたものは、従来の準備金計算モデルと、機械学習・深層学習的手法を適用した準備計算モデルの比較でした。そのため、検討項目は、データの選定から、データの前処理ですね。そちらを経たあと、従来のモデルおよび機械学習・深層学習モデルの構築、そして、最終的に結果の比較。そのような一連の分析過程を網羅したものとなりました。本発表では、その分析過程を最初から最後まで、一気通貫した形でご報告させていただきます。そのため、容量が多いので、非常に駆け足なところもあるとは思いますが、ご承知いただけましたらと思います。

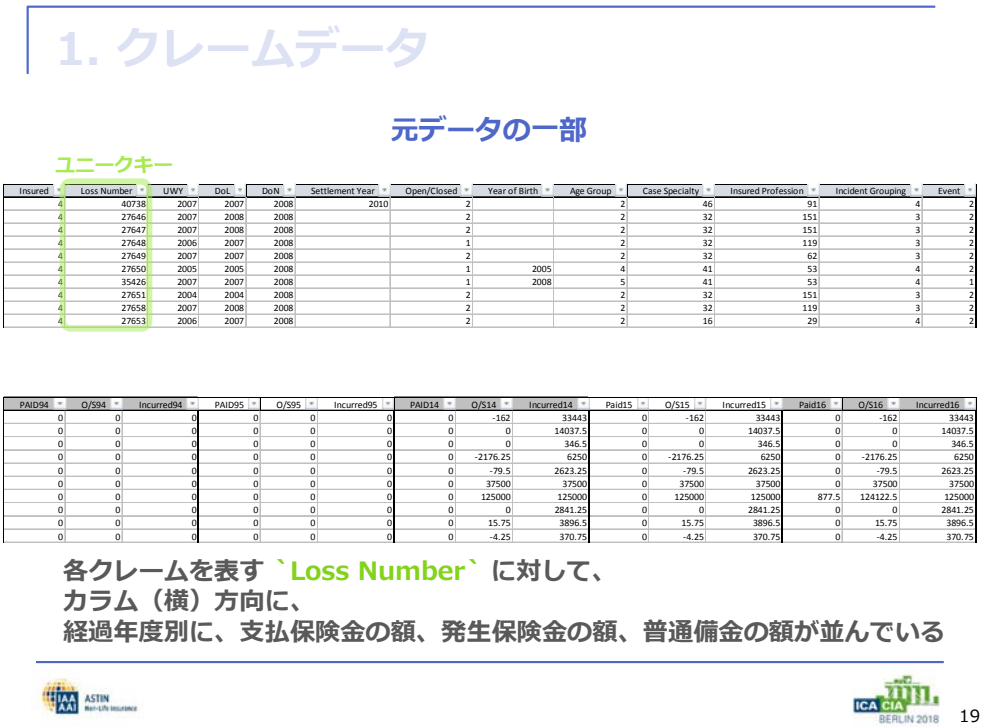


今、お話しいたしましたとおり、一連の分析過程を 34 名のメンバー間で分担いたしましたが、私の担当項目は、先ほどからお見せいたしております、このパイプラインといいますか、一連のフローなのですけれども、こちらの部分になります。私が担当しましたものは、主に機械学習・深層学習モデル用の validation の枠組みの設計と、基礎コードの構築でした。こちらの validation という言葉は、お聞きになられない方も多数いらっしゃると思いますので、後ほどご説明をさせていただきます。また、本ワーキングパーティーの分析過程は分担制でしたので、私の担当部分以外で非常に細かいところになってきますと、なぜそうなったのかなどの経緯も含めまして、あまり定かではない部分もあるため、説明が少しあやふやな部分はあるかもしれませんが、ご承知いただけましたらと思います。



それでは、検討内容の詳細の説明に入らせていただきます。こちらは、先ほどからお見せいたしております、検討内容の一連の流れを図示したパイプラインなのですけれども、分かりやすく明滅させていると思いますが、従来の準備金モデルと機械学習・深層学習モデルで使用するデータの形態は、非常に大きく異なります。従来の準備金モデルで使用しておりますデータは、事故年度・経過年数別に集積されたトライアングルデータの形態を取ります。一方で、個々のクレームデータごとの粒度になっているものが、機械学習・深層学習モデルとなります。こちらのデータは、validation 用にですね、train、valid、test と、３つにデータを分割して使用することになります。

それでは、一連のプロセスを、最初から説明に入らせていただきたいと思います。まず元データに関して、ご説明をいたします。元データなのですけれども、Swiss Re 様から、ご厚意で、専門職業人賠償責任保険のデータを提供いただきました。こちらは、私も聞き慣れなかったのですけれども、基本的には労災補償の一種とお考えいただければと思います。データの取得期間は、1994 年から 2016 年までとなります。また、契約は、2009 年度までにされたものとなっていますので、以降は run off ですね。経過のみを観察したデータとなっています。データはクレームごとの粒度となっておりまして、スライド内の表にありますような変数を持っております。主に経過年数ごとの支払保険金、発生保険金、普通備金。また、Underwriting year などの、時期に関連した項目ですね。タイミングですね。そして、３つ目が、契約者の年齢や、クレームの性質などに関連した項目。以上、大別すると３つから成っています。



Insured	Loss Number	UWY	DoL	DoN	Settlement Year	Open/Closed	Year of Birth	Age Group	Case Specialty	Insured Profession	Incident Grouping	Event
4	40738	2007	2007	2008	2010	2		2	46	91	4	2
4	27646	2007	2008	2008		2		2	32	151	3	2
4	27647	2007	2008	2008		2		2	32	151	3	2
4	27648	2006	2007	2008		1		2	32	119	3	2
4	27649	2007	2007	2008		2		2	32	62	3	2
4	27650	2005	2005	2008		1	2005	4	41	53	4	2
4	35426	2007	2007	2008		1	2008	5	41	53	4	1
4	27651	2004	2004	2008		2		2	32	151	3	2
4	27658	2007	2008	2008		2		2	32	119	3	2
4	27653	2006	2007	2008		2		2	16	29	4	2

PAID94	O/S94	Incurred94	PAID95	O/S95	Incurred95	PAID14	O/S14	Incurred14	Paid15	O/S15	Incurred15	Paid16	O/S16	Incurred16
0	0	0	0	0	0	0	-162	33443	0	-162	33443	0	-162	33443
0	0	0	0	0	0	0	0	14037.5	0	0	14037.5	0	0	14037.5
0	0	0	0	0	0	0	0	346.5	0	0	346.5	0	0	346.5
0	0	0	0	0	0	0	-2176.25	6250	0	-2176.25	6250	0	-2176.25	6250
0	0	0	0	0	0	0	-79.5	2623.25	0	-79.5	2623.25	0	-79.5	2623.25
0	0	0	0	0	0	0	37500	37500	0	37500	37500	0	37500	37500
0	0	0	0	0	0	0	125000	125000	0	125000	125000	877.5	124122.5	125000
0	0	0	0	0	0	0	0	2841.25	0	0	2841.25	0	0	2841.25
0	0	0	0	0	0	0	15.75	3896.5	0	15.75	3896.5	0	15.75	3896.5
0	0	0	0	0	0	0	-4.25	370.75	0	-4.25	370.75	0	-4.25	370.75

次に、実際のデータの一例をごらんいただきたいと思います。元データなのですけれども、各クレームのユニークキーである Loss Number というものが、カラムの名前として元々ございまして、それに対して横方向に、経過年度別に支払保険金の額、発生保険金の額、普通備金の額が並んでいます。このようなテーブルデータの形を、「ワイドフォーマット」と呼びます。データ点数は、このワイドフォーマットで約５万点ほどでした。後ほどお話をいたしますが、こちらのテーブルを、各クレームに対して経過年数ごとに横だったものを、縦に、行方向に分解した形であります「ロングフォーマット」という形式にいたします。その場合ですと、当然データ点数は増えるのですけれども、61 万点ほどのデータ点数となります。

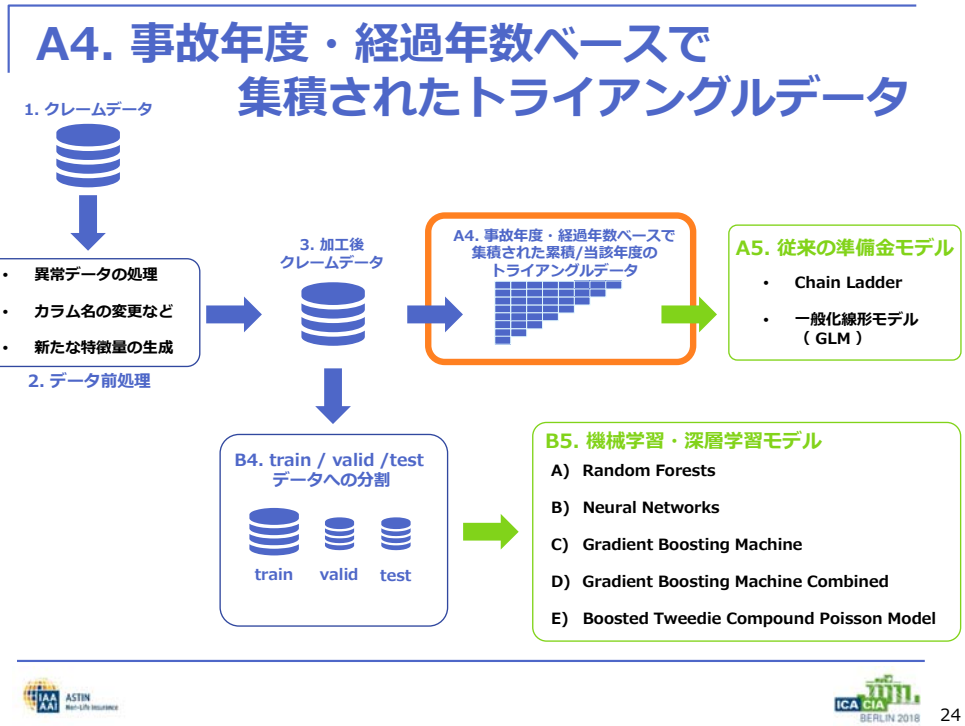
では、今、お話しいたしましたようなワイドフォーマット、ロングフォーマットも含めまして、データの前処理に関して、ご説明をいたします。データの前処理なのですけれども、先ほどお話しいたしましたように、列方向に経過年数ごとの支払保険金、発生保険金、普通備金が並んでいるワイドフォーマットから、行方向に経過年数ごとにそれらが並んでいるようなロングフォーマットへと変換いたします。これは、同じクレームでも、経過年数分のデータ点数を持つということを意味いたしますので、非常にデータ点数が増えることになります。また、例外データの処理も行っております。例えば、1994 年よりも前にクレームが発生していて、いまだにクローズしていないものを除外したり、クレームの報告がクレームの発生よりもなぜか前になっているような、多分、何かのミスだとは思うのですけれども、そのような異常なデータを除外したりしています。

また、前処理で一番重要になってきますものが、新たな特徴量の生成です。会場の皆様の中には、特徴量という言葉が耳慣れない方もいらっしゃるとは思うのですけれども、データサイエンスの分野では非常に一般的な言葉でして、基本的には説明変数と同義、同じ意味だとお考えいただければと思います。今後は、本発表ではこちらの言葉を使用させていただきます。

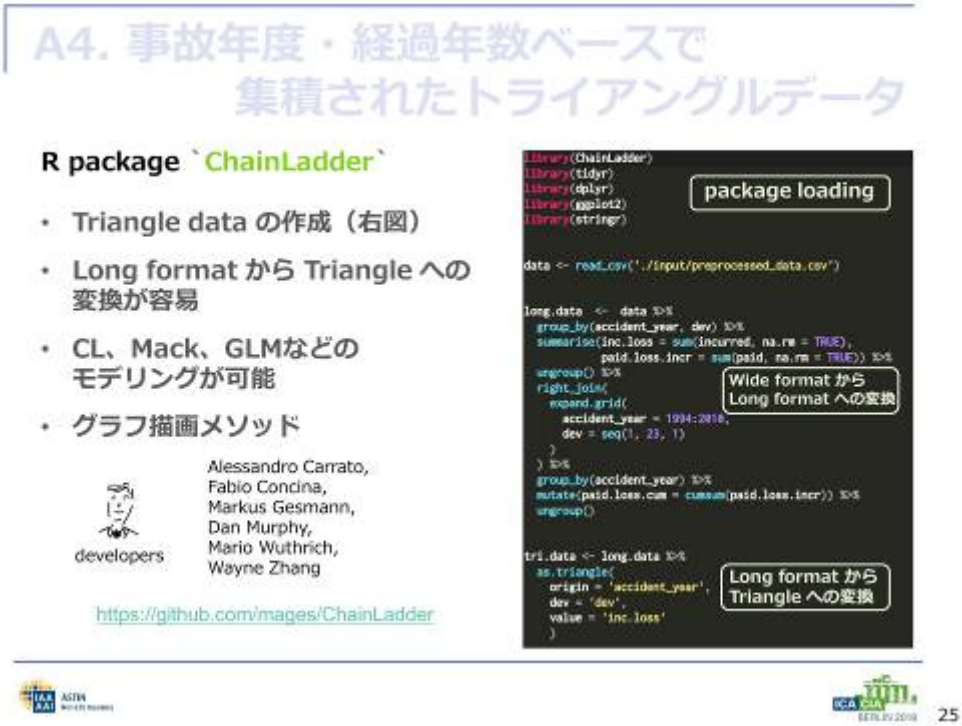
また、新たに作成します特徴量なのですけれども、こちらのスライドにありますように、calendar_year や report_lag など、簡単に四則演算で求められるような特徴量を作ることになります。これは、後ほど少し言及できるかもしれないのですけれども、基本的に機械学習や深層学習のモデルは、簡単な四則演算で作れる特徴量をなかなかモデル内部で作りにくい、自分たちで表現しにくいところがありまして、そのため、前もって前処理の段階でそのようなところを作ることになります。このようにして作成した特徴量は、後ほどご説明させていただきますが、いろいろな機械学習・深層学習のモデルで使うことになります。

以上のプロセスを経まして、加工されたクレームデータができあがります。ここでは、そのデータの一部をごらんいただきます。こちらがデータの一例なのですけれども、クレームデータのユニークキーであります claim_number が、1000 ですね。少し見にくいかもしれないのですけれども、左上が claim_number なのですが、全部 1000 だと思います。その 1000 のデータだけを取り出してきて、表示したものになります。先ほどからご説明させていただいておりますが、行方向に、経過年数ごとにこちらのクレームの累計が並ぶデータとなっています。

そして、左下の緑枠で囲んでいる、少し見にくいかもしれませんが、こちらですね。こちらが accident_year に対して calendar_year を示しているのですけれども、claim_number が 1000 のものは accident_year が 2000 年なのですけれども、calendar_year が、2000 年から 2001 年、2002 年と、2016 年まで並んでいるということで、ロングフォーマットにきちんとなっていることがお分かりいただけるかと思います。



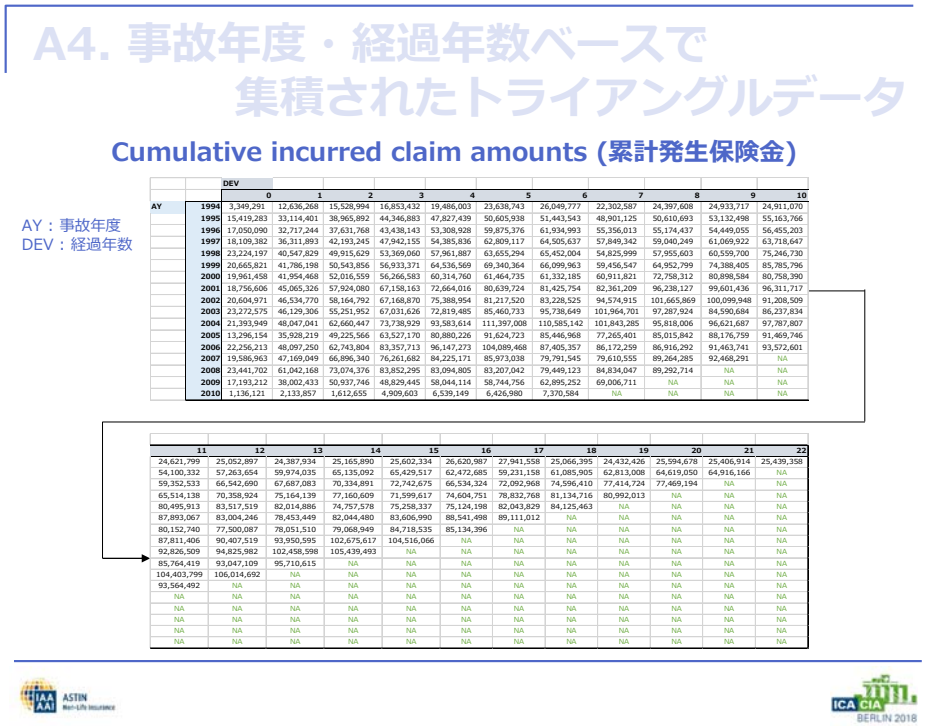
それでは、データの前処理が終わりましたので、これから、従来の伝統的手法、機械学習・深層学習的手法の説明に移らせていただきたいと思います。まず、従来の準備金モデルに関しまして、こちらは、チェインラダーに代表されますように、トライアングルデータを使用いたします。これまでデータの加工を通して得られておりますものは、ロングフォーマットのデータですので、そのデータをチェインラダー等に適した形に集計をいたします。



トライアングルデータを作成するに当たりまして、使っていたものはRなのですけれども、Rパッケージで「ChainLadder」という非常に有用なパッケージがございます。こちらを使用することで、ロングフォー

マットからトライアングルフォーマットへと簡単に変換することができます。また、こちらのパッケージなのですが、チェインラダー、Mack モデル、GLM といった各種モデリング手法も、パッケージの中にそのような関数がございますので、使用することができ、従来のモデリング手法に関しては、非常に充実したパッケージとなっています。開発者ですけれども、Wuthrich など、有名なリザービング・アクチュアリーですね。そのような方々が開発されているので、パッケージとしての信頼度も、それなりに高いのではないかと個人的に考えています。

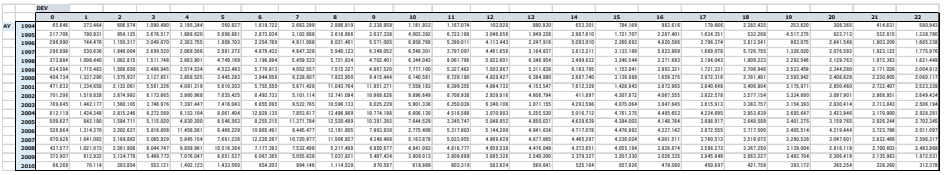
右側の図にありますものは、コードの全文ですね。一例ではなく、全文です。一番上段でパッケージをロードして、二段目でワイドフォーマットからロングフォーマットに変換する。そして、最後、ロングフォーマットからトライアングルデータへの変換は、チェインラダーパッケージの「as.triangle」という関数を使うと一発でできるということで、ほとんど苦労することなく、このようなデータテーブルの集計ができます。



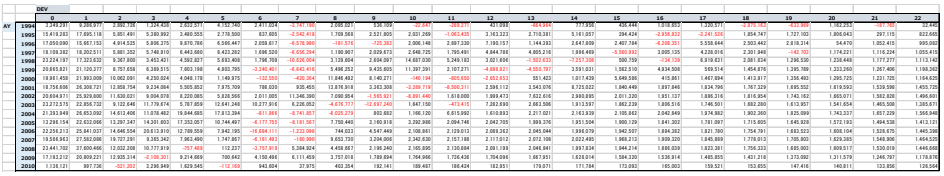
そのようにできたものが、こちらは累計支払保険金のトライアングルデータなのですけれども、エクセルにペーストしただけなのですが、このような形で集計ができます。単位は、アメリカドルとなっています。観察データがあるものは、最終 calendar_year は 2016 年なのですけれども、見切れてしまっているので上と下に分割していますが、ご了承ください。この対角上が、calendar_year 2016 年に該当するという形になります。縦軸・横軸は、事故年度・経過年数という形になります。そして、先ほどが累計支払保険金ですが、同じように、累計発生保険金のトライアングルデータも簡単に作ることができます。

A4. 事故年度・経過年数ベースで集積されたトライアングルデータ

Incremental paid claim amounts (当該年度 支払保険金)



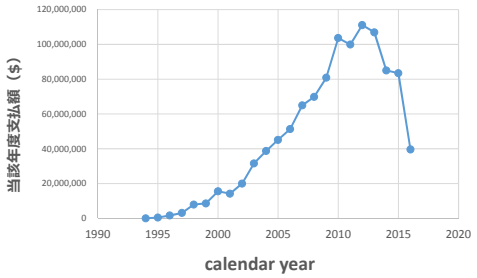
Incremental incurred claim amounts (当該年度 発生保険金)



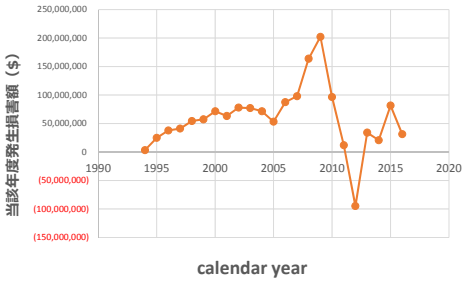
また、先ほどは Cumulative な類型に関してのデータでしたけれども、こちらは Incremental な、当該年度と書いていますけれども、そちらの支払保険金と発生保険金のデータになります。特徴的なことは、発生保険金のデータが、赤字のセルが何か所か散見されると思うのですけれども、発生保険金がマイナスであることを意味いたしております。残念ながら、Swiss Re 様に理由までは聞けなかったのですけれども、何らかの形でこのような異常なことが、しばしばあるとは思うのですが、実験データ上も出てきてしまっているということになります。

A4. 事故年度・経過年数ベースで集積されたトライアングルデータ

Incremental paid claim amounts (当該年度 支払保険金)



Incremental incurred claim amounts (当該年度 発生保険金)



1. **2009年までの契約データが対象のため、2012年以降は支払額が減少**
2. **当該年度の発生保険金は、volatility が大きい（少し特殊なケース）**

以上のトライアングルデータに対しまして、calendar year ごとに支払保険金および発生保険金を集計し

た結果が、こちらのグラフとなります。最終累計支払保険金と最終発生保険金ですね。ごめんなさい、当該年度の支払保険金ですね。2009 年度までの契約データが対象ですので、支払保険金は、2009 年を過ぎますと、2012 年以降くらいから減少傾向にあります。一方で、発生保険金ですけれども、トライアングルデータでもマイナスの場所があるとお話しさせていただきましたが、そちらを受けて、非常に変動幅の大きい、不安定なカーブを描きます。こちらは、後ほどスライドの最後の方のモデルの検証結果でご説明させてはいただくのですが、支払保険金は安定した傾向にあって、モデリングがしやすい一方で、発生保険金は、ごらんのとおり不安定ですので、モデリングはそれほど簡単ではないデータになっています。

A5. 従来の準備金モデル

Chain Ladder（チェインラダー法）

1. **累計（支払 / 発生）保険金に基づいた LDF（ロスディベロップメントファクター）を計算し、将来・最終累計（支払 / 発生）保険金を算出。**

2. **各経過年数毎の LDF を算出するにあたり、AY 2010 年のトライアングルデータは、2009 年の written premium のうち 2010 年に earned となった契約に関するものであり、データ数が少ないため、計算対象から除外。**

3. **LDF の計算対象セルを、異常な値を除くように選択。**

それでは、従来の準備金モデルの説明に入りたいと思います。本ワーキングパーティーでは、チェインラダーと一般化線形モデルの２つを検討いたしました。まずはチェインラダーに関しまして、簡単にご説明させていただきます。チェインラダーに関しましては、トライアングルデータの累計支払・発生保険金に対しまして、ロスディベロップメントファクターを作り、それを基に、将来および最終累計支払・発生保険金を算出いたしました。

また、各経過年数ごとのロスディベロップメントファクターを算出するに当たりまして、事故年度が 2010 年度のデータは除外いたしております。その理由といたしましては、契約自体が 2009 年度までのものしかなかったので、2010 年度に事故となったものは 2009 年度契約のもので、データ点数が非常に少なく、信頼性に欠けるものであったためということになります。また、同時に、これは非常に主観的なので、いろいろ議論はあるとは思うのですけれども、ロスディベロップメントファクターが非常に異常な値となっていたものは、分析者の方で幾つか除外をいたしております。

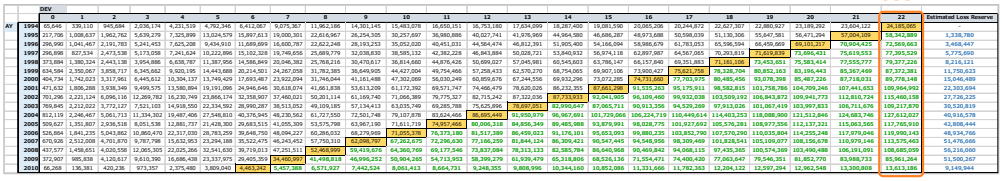
では、次のページで、トライアングルデータに対して、どのセルをモデルのフィッティングに使用して、どのセルを予測したかをごらんいただきたいと思います。こちらの考え方は、後々機械学習・深層学習のモデルと比較する際にバックテストを行いますが、そちらにおいて非常に重要な考え方となってきます。

A5. 従来の準備金モデル

モデルのフィッテング と 評価手法

CY 2016

Ultimate Loss

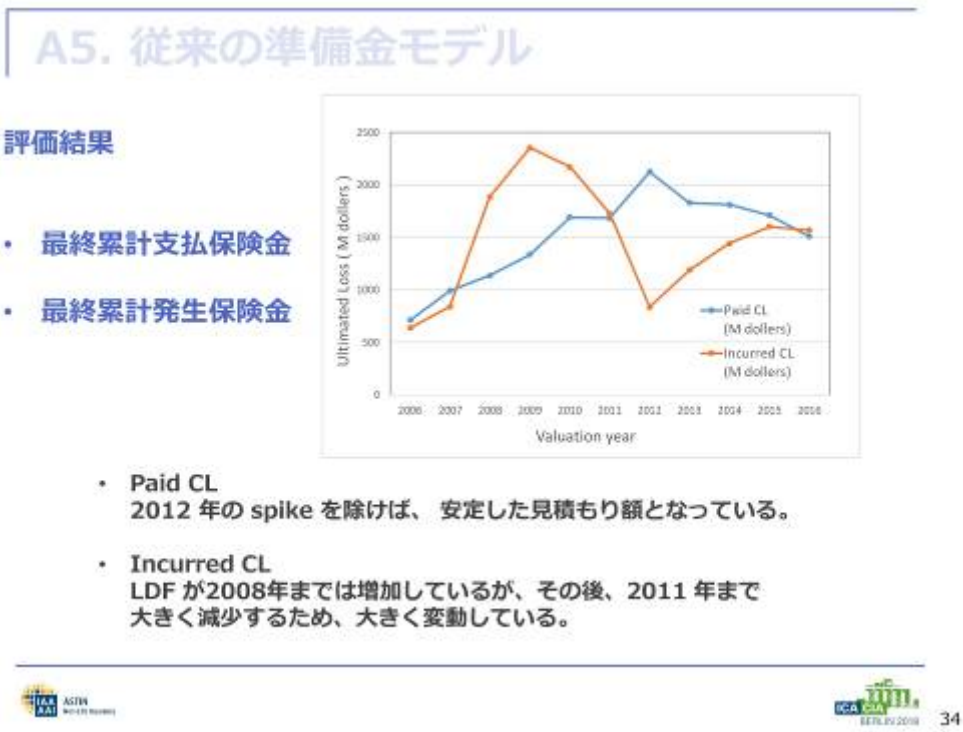


AY	DEV 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Estimated Loss Reserve
1994	65,646	339,110	945,684	2,036,174	4,231,519	4,792,346	6,412,067	9,075,367	11,962,186	14,301,145	15,483,078	16,650,151	16,753,180	17,634,099	18,287,400	19,081,590	20,065,206	20,244,872	22,627,307	22,880,927	23,189,292	23,604,122	24,185,065	
1995	217,706	1,008,637	1,962,762	5,639,279	7,325,899	13,024,579	15,897,013	19,000,301	22,616,967	26,254,305	30,257,697	36,980,886	40,027,741	41,976,969	44,964,580	46,686,287	48,973,688	50,598,039	51,130,306	55,647,581	56,471,294	57,004,109	58,342,889	1,338,780
1996	296,990	1,041,467	2,191,783	5,241,453	7,625,208	9,434,910	11,689,699	16,600,787	22,622,248	28,193,253	35,052,020	40,451,031	44,564,474	46,812,391	51,905,400	54,166,094	58,986,679	61,783,053	65,596,594	66,459,665	69,181,317	70,904,429	72,569,063	3,468,447
1997	296,898	827,534	2,473,538	5,173,058	7,241,624	10,222,896	15,102,328	19,749,656	25,689,779	32,038,830	38,585,132	42,382,228	46,843,884	50,028,721	53,840,932	56,974,118	62,897,987	64,567,065	70,293,819	71,819,835	73,695,431	75,619,553	77,395,529	5,775,690
1998	373,694	1,380,324	2,443,138	3,954,886	6,638,787	11,387,956	14,586,849	20,046,382	25,768,216	30,470,617	36,814,660	44,876,426	50,699,027	57,045,981	60,545,603	63,786,147	66,157,840	69,351,883	71,160,166	73,453,651	75,383,414	77,555,777	79,377,226	8,216,121
1999	634,584	2,350,067	3,858,717	6,345,662	9,920,195	14,443,688	20,214,561	24,267,058	31,782,385	36,649,905	44,427,004	49,754,466	57,258,433	62,570,270	68,754,065	69,907,106	73,900,427	76,621,798	78,128,704	80,852,163	83,196,443	85,367,469	87,372,381	11,750,623
2000	404,734	1,742,023	3,317,961	6,445,612	10,394,137	13,749,429	17,693,487	23,922,094	31,746,044	41,161,488	47,302,069	56,030,249	60,859,676	67,244,556	69,932,296	73,072,285	74,731,660	77,703,975	80,485,456	83,078,398	85,487,226	87,718,031	89,778,148	15,046,489
2001	471,632	1,806,288	3,938,349	9,499,575	13,580,894	19,191,896	24,946,640	30,618,074	41,661,838	53,613,269	61,172,392	69,571,747	74,466,479	78,620,026	84,232,355	87,661,298	91,535,363	95,175,911	98,582,815	101,758,766	104,709,246	107,441,453	109,964,992	22,303,694
2002	701,296	2,221,124	6,096,116	12,269,762	16,230,749	23,866,174	32,358,907	37,460,021	50,201,114	61,169,740	71,066,389	79,775,327	82,715,242	87,322,036	87,773,931	92,041,902	96,109,469	99,932,038	103,509,192	106,841,872	109,941,773	112,810,724	115,460,158	27,726,225
2003	769,845	2,212,022	3,772,127	7,521,103	14,918,552	22,334,592	28,990,287	38,513,052	49,109,185	57,134,413	63,035,749	69,285,788	75,625,896	78,697,551	82,990,647	87,065,711	90,913,356	94,529,269	97,913,026	101,067,419	103,997,833	106,711,676	109,217,970	30,520,819
2004	812,119	2,246,467	5,061,713	11,334,302	19,487,406	27,548,810	40,376,945	46,230,562	61,727,550	72,501,748	79,107,878	83,624,466	86,685,449	91,850,679	96,967,691	101,729,066	106,224,719	110,446,614	114,403,253	118,088,900	121,512,846	124,683,746	127,612,027	40,916,578
2005	509,627	1,351,807	2,936,518	8,051,538	12,881,737	21,428,300	29,683,515	41,055,309	53,575,798	63,967,190	71,611,719	74,957,468	80,006,318	84,856,349	89,485,988	93,879,991	98,038,775	101,927,692	105,576,281	108,977,556	112,137,321	115,063,565	117,765,910	42,808,444
2006	526,864	1,841,235	5,043,862	10,860,470	22,317,030	28,783,259	39,648,755	48,094,227	60,286,032	68,279,969	71,955,376	76,373,188	81,517,389	86,459,023	91,176,101	95,653,093	99,880,235	103,852,790	107,570,290	110,835,804	114,255,248	117,979,046	119,990,143	48,934,766
2007	670,926	2,512,008	4,701,870	9,787,798	15,632,953	23,294,188	35,522,475	46,243,452	57,750,310	62,066,797	67,362,675	72,296,838	77,166,259	81,844,124	86,309,421	90,547,445	94,548,956	98,309,469	101,828,541	105,189,877	108,156,678	110,979,146	113,575,463	51,476,666
2008	437,577	1,498,651	4,020,558	12,065,305	22,025,266	32,541,630	39,719,011	47,251,511	52,468,999	59,419,676	64,360,769	69,177,546	73,837,084	78,313,133	82,585,784	86,640,968	90,469,842	94,068,115	97,435,365	100,574,369	103,490,488	106,191,091	108,685,059	56,216,060
2009	372,907	985,838	4,120,617	9,610,390	16,686,438	23,337,975	29,405,359	34,480,997	41,498,818	46,996,252	50,904,265	54,713,953	58,399,279	61,939,479	65,318,806	68,526,136	71,554,471	74,400,426	77,063,647	79,546,351	81,852,770	83,988,733	85,961,264	51,590,267
2010	66,268	136,381	420,236	973,357	2,375,488	3,809,040	4,483,342	5,457,388	6,571,927	7,442,534	8,061,413	8,664,731	9,248,355	9,808,996	10,344,160	10,852,086	11,331,666	11,782,363	12,204,122	12,597,294	12,962,548	13,300,808	13,613,186	9,149,944

- **白色セルと黄色セルは、モデルのフィッテングに使用するデータ**
- **黄色セルは、Calendar Year (CY) = 2016 時点のセルに該当**
- **緑色セルが予測すべきセル**

本ワーキングパーティーでは、Calendar Year（CY）によって、フィッティングに使用するセルと予測すべきセルを分けております。例えば、こちらのスライドの図ですと、CY が 2016 年までのデータをフィッティング用のセル、それ以降を予測すべきセルとした場合のものになります。白色と黄色のセルが、フィッティングに使用しますデータです。緑色の文字のセルが、予測すべきセルになります。これはもう予測後のものなので、埋まってしまっているのですけれども、そこが予測すべきセルになります。黄色のセルは、対角上にあることからも、CY が 2016 年のものとお分かりいただけるかと思います。こちらは全て、先ほどお話しいたしましたとおり、累計ベースのデータで、こちらは支払保険金のものになります。同様に、発生保険金に関してもモデリングをすることになります。

以上のように、2016 年から 2006 年にかけて、CY を基準としまして、トライアングルデータをフィッティングと予測用に分けまして予測してみた結果が、次のページのようになります。ここでは、モデルの当てはまり具合、バックテストに関してはご紹介しません。後ほど、最後の機械学習・深層学習的モデルと併せて、ご説明をさせていただきます。



こちらが、チェインラダーの最終累計支払保険金と最終累計発生保険金を図示したものとなります。このグラフのｘ軸、「Valuation year」とありますけれども、こちらが、トライアングルデータのセルをフィッティング用と予測用に分ける時の基準となっています、CY のことです。「Calendar year」と書けばよかったのですけれども、すみません。こちらのグラフから読み取れますが、支払保険金の最終累計は、2012 年にちょっとしたスパイクがあるのですけれども、そちらを除けば、大体収束していく、安定した見積もりとなっています。

一方で発生保険金は、非常に大きく変動しています。これが非常にモデリングしにくい理由になるのですが、実績値として、2010 年以降に発生保険金が大きくマイナスとなる年が、前のスライドにあったと思います。それらを考慮したロスディベロップメントファクターを使って計算すると、このように非常にボラティリティーの激しいカーブになってしまいます。以上が、チェインラダーの結果の説明になります。

A5. 従来の準備金モデル

GLM（一般化線形モデル）

1. 当該年度（支払 / 発生）保険金（incremental）を、
   GLM で予測する目的変数とした。

2. 当該年度支払保険金の GLM には、累積分布関数の
   あてはまりの良さから、log リンク関数とガンマ分布を採用。

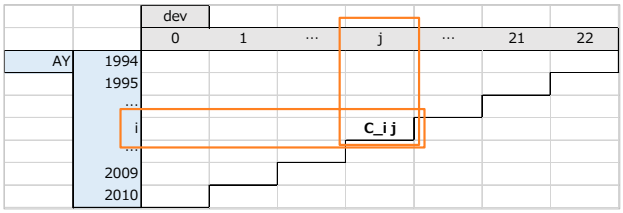
3. 当該年度発生保険金の GLM には、負の実績値が存在したため、
   恒等リンク関数と正規分布を採用。

```
library(ChainLadder)
library(tidyverse)
library(ggplot2)
library(stringr)


load('./input/tri_paid.RData')
load('./input/tri_inc.RData')
glm.paid <- glmReserve(triangle = tri.paid, var.power = 2, link.power = 0)
glm.inc <- glmReserve(triangle = tri.inc, var.power = 0, link.power = 1)
```


次に、GLM に関して説明をさせていただきます。GLM の場合は、チェインラダー法と異なり、当該年度の支払保険金・発生保険金を予測する形となります。先ほどが Cumulative なのに対して、今回は Incremental でやるという形になります。支払保険金の分布には、log リンク関数とガンマ分布を採用いたしました。一方で発生保険金の分布には、先ほどからご覧いただいておりますが、マイナスの値ですね。負の値がございますので、恒等リンク関数と正規分布を採用いたしました。下の図は、R のパッケージ、先ほどご紹介した ChainLadder ですね。そちらを使用いたしまして、incremental なトライアングルデータに対して GLM を適用するコードとなります。ものの２行で終わってしまうのですね。非常に楽です。簡素なコードでできるということになります。

A5. 従来の準備金モデル

当該年度支払保険金 $C_{i,j}$ の場合



- 事故年度と経過年数をリンク関数の説明変数とした GLM

$$E(C_{i,j}) = \mu_{i,j}$$
$$\ln(\mu_{i,j}) = \mu + \alpha_i + \beta_j$$

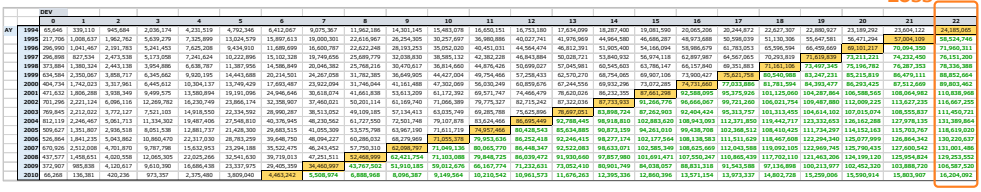
α_i：事故年度 i に対応する係数
β_j：経過年数 j に対応する係数

GLM におけるリンク関数のご説明を、もう少しさせていただきます。本検討では、個別クレームごとに GLM を組み立てているわけではなく、トライアングルデータに対してやっておりますので、集積されてしまっているわけですね。ですので、集積された値が持つ特徴量といいますか、そのようなものは非常に限られておりまして、結局、アクシデントイヤーとディベロップメントイヤー、つまり経過年数ですね。そちらを２つの説明変数として、リンク関数とひもづけています。

A5. 従来の準備金モデル

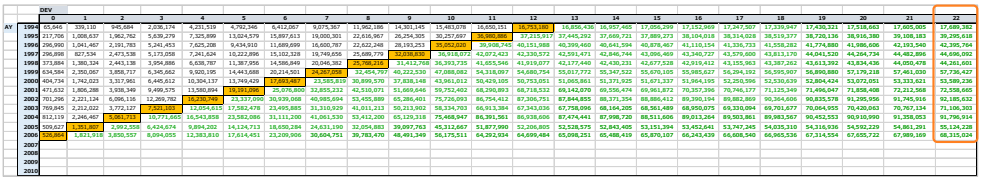
CY 2016

Ultimate Loss



CY 2006

Ultimate Loss



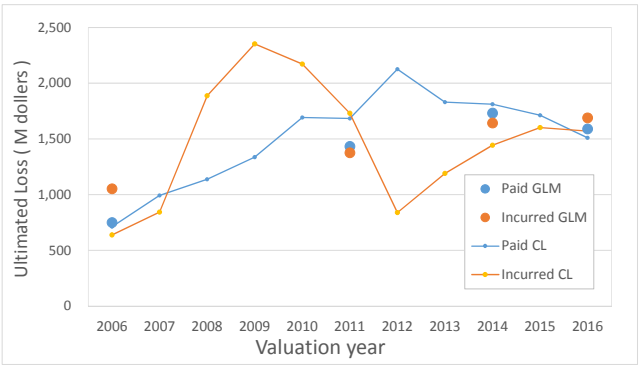
以上の前提を基に作成いたしました GLM の結果は、一応、トライアングルデータで見ると、こちらのような形になります。１点、チェインラダーにおける検討と異なっておりますところは、2006 年から 2016

年、全部の CY に対してバックテストのような検討を行ったわけではなくて、2006 年、2011 年、2014 年、2016 年と、４つの年度だけ選んでやる形になっています。これは、先ほどチェインラダーの分析結果をお見せしたと思うのですが、その年度以外だとかなり予測がぶれぶれだというところから、比較的安定した年度を恣意的に選んでいます。

A5. 従来の準備金モデル

評価結果

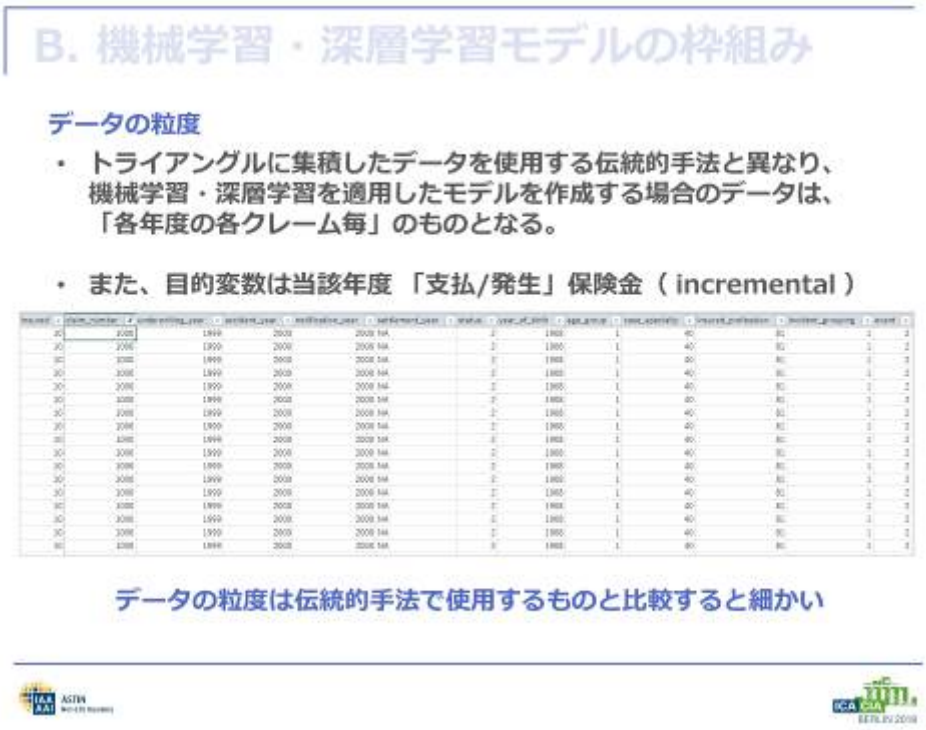
- **最終累計支払保険金**
- **最終累計発生保険金**



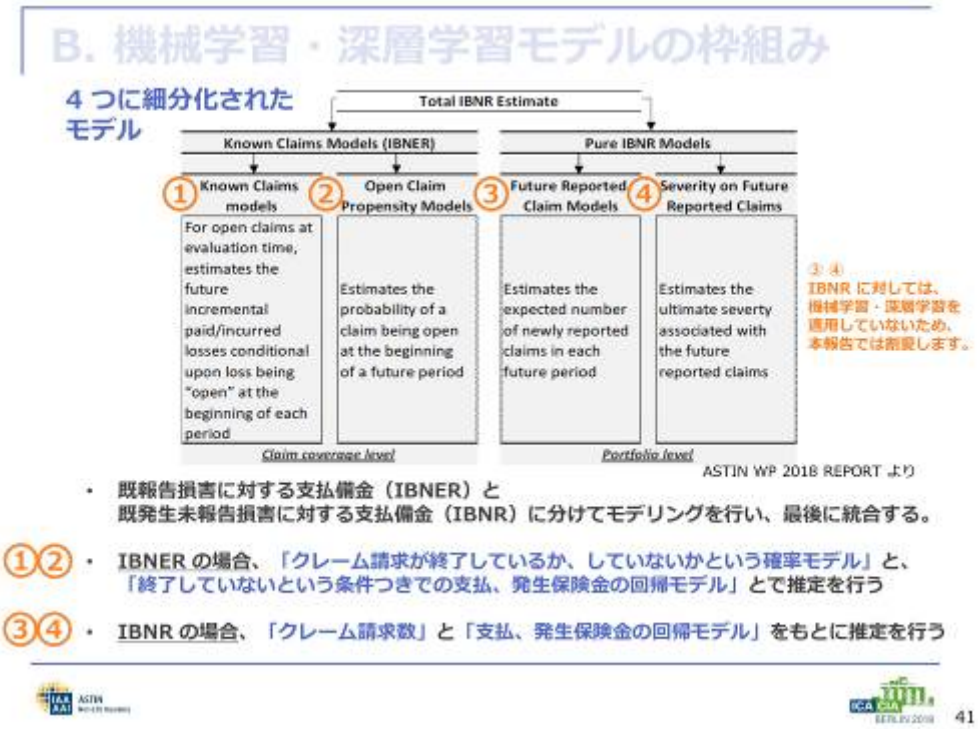
- **本 WP では、GLM の計算に関しては、valuation year が、2006, 2011, 2014 及び 2016 年の場合のみ実施。（後述する機械学習・深層学習的手法においても同様）**
- **4 種の valuation year において、CL と全体的な傾向は似通っている。**

次に、チェインラダーとの比較結果をごらんいただこうと思います。こちらが、チェインラダーの結果と GLM の結果をまとめたものとなります。両モデルの最終累計支払保険金と最終累計発生保険金ですけれども、両モデルの 2006 年、2011 年、2014 年、2016 年。こちらに関しましては、GLM もチェインラダーも、比較的似通ったような傾向にあることがごらんいただけるかと思います。

本スライドまでが、従来手法の説明となります。これ以降は、機械学習・深層学習的手法の説明に移らせていただきます。まずはじめに、これらのモデルで共通に利用いたしますフレームワーク、validation ですね。先ほどからお話しいたしておりました。そちらに関しまして、説明をさせていただきます。



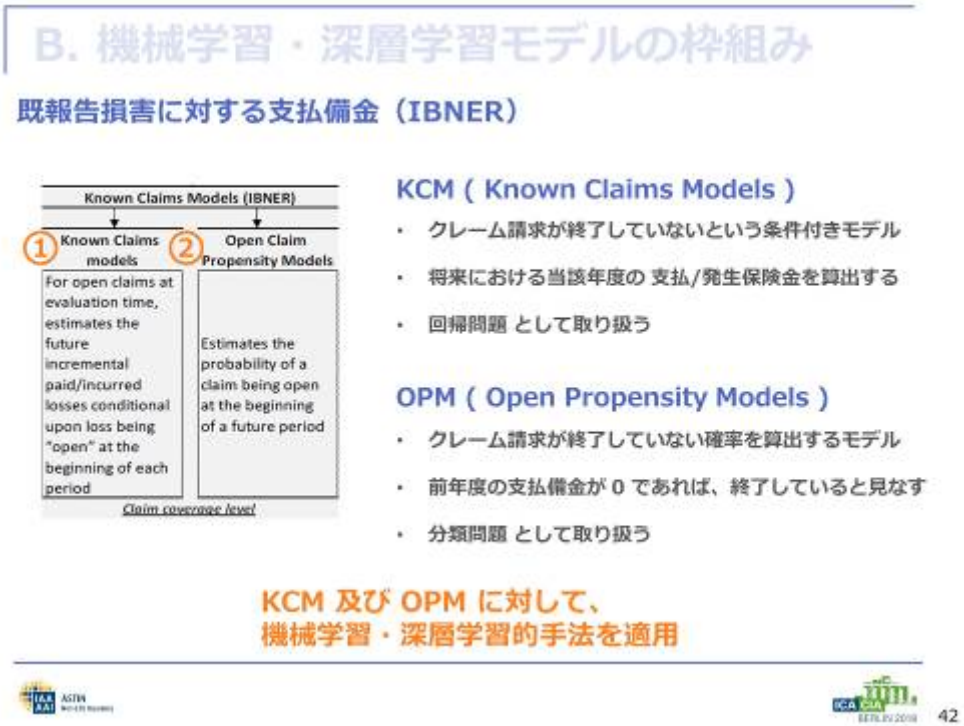
ごめんなさい。その前に、データの粒度に関してご説明させていただきます。使用するデータの粒度なのですけれども、先ほどご説明いたしましたとおり、トライアングルデータとは異なります。クレームごと、かつ経過年数ごとの、粒度の細かいものとなります。また、機械学習・深層学習モデルが予測する対象としますのは、GLM と同様に、累計ベースではなく、incremental な当該年度の支払保険金・発生保険金となります。



では、先ほど言いかけましたフレームワークに関して、ご説明をさせていただきます。支払備金のモデルなのですけれども、２つに大きく分けてモデリングすることになります。１つ目が、既に報告済みのク

レームに対する支払備金、IBNER 対するモデルとなります。このモデルを、本ワーキングパーティーでは「Known Claims Models」と呼称しておりました。こちらを、次のスライド以降でご説明いたしますが、KCM および OPM というモデルに分けることになります。KCM は、クレーム請求が終了していないという条件付きでの支払保険金・発生保険金の値を回帰するモデルになっておりまして、一方で OPM は、クレーム請求が終了しているか、終了していないかという確率を表すモデルとなります。

２つ目は、発生はしているのですが、未報告のクレームに対する支払備金、IBNR に対するモデルとなります。こちらも少し変わった呼び方で、一般的なのでしょうか？ピュア IBNR と呼んでおりまして、クレーム数とクレーム損害額を基にした Frequency-Severity model、FD 法のモデルになります。こちらに関しましては、ワーキングパーティーもなかなか締め切りまでの時間があまりなく厳しかったものですから、機械学習・深層学習の手法を適用しておりません。ですので、本発表では、こちらの報告は割愛させていただきます。とは言いましても、最終的な予測値を算出する場合には、以上の２つのモデルの予測値を合算する形となります。

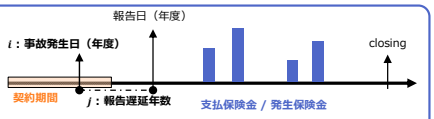


こちらのスライドでは、もう少し詳しく、既報告損害に対します支払備金の予測モデルの説明をさせていただきたいと思います。先ほどの KCM に関しましては、クレーム請求が終了していないという条件付きのもとで、将来における当該年度の支払/発生保険金を算出することになります。発生保険金そのものを予測するモデルですので、KCM は回帰問題となります。

一方で OPM に関しましてですけれども、こちらは、クレーム請求が終了しているか、していないかという確率を算出するモデルですので、二値の分類問題として取り扱うこととなります。ただ、教師データとして使います実績値として、クレーム請求が終了しているか、していないかというものに関しましては、前年度の備金がゼロであれば終了していると見なすというルールを設けて、データ上で判断いたしております。こちらに関しましては、当然例外もあるかとは思うのですけれども、例えばリオープンするなど、いろいろなケースもあると思うのですが、データから得られている情報が限られておりますため、そのような形で判断いたしております。

B. 機械学習・深層学習モデルの枠組み

i：事故年度
j：報告遅延年数
N_{ij}：事故年度 i 、報告遅延年数 j におけるクレームの数
$\mathcal{F}_t$：t 時点で判明している情報
v：各クレーム
$C_{i,j|k}^{(v)}$：事故年度 i 、報告遅延年数 j 、クレーム v の 経過年数 k における（支払/発生）保険金



事故年度 i の t 年度における最終累計（支払/発生）保険金は、

$$\sum_{j=0}^{t-i}\sum_{v=1}^{N_{ij}}\left(\sum_{k=0}^{t-(i+j)} C_{i,j|k}^{(v)} + \sum_{k>t-(i+j)} E\left[C_{i,j|k}^{(v)} \mid \mathcal{F}_t\right]\right)$$

とかける。（Wuthrich 2016）

■ ここで、KCM 及び OPM の考え方にそって上式を拡張した、既発生クレームの最終累計（支払/発生）保険金は、

$x_{i,j|k}^{(v)}$：事故年度 i 、報告遅延年数 j 、クレーム v の 経過年数 k におけるクレームの特徴量
$OPM_{j+k}\left(x_{i,j|k}^{(v)}\right)$：事故年度 i 、報告遅延年数 j 、経過年数 k のクレーム v の請求が終了していない確率
$KCM_{j+k}\left(x_{i,j|k}^{(v)}\right)$：事故年度 i 、報告遅延年数 j 、経過年数 k のクレーム v の請求が終了していない条件付き $i+j+k$ 年度におけるクレームの期待（支払/発生）保険金

事故年度 i の t 年度における最終累計（支払/発生）保険金は、

$$\sum_{j=0}^{t-i}\sum_{v=1}^{N_{ij}}\left(\sum_{k=0}^{t-(i+j)} C_{i,j|k}^{(v)} + \sum_{k>t-(i+j)} KCM_{j+k}\left(x_{i,j|k}^{(v)}\right)\times\frac{OPM_{j+k}\left(x_{i,j|k}^{(v)}\right)}{OPM_{j+k}\left(x_{i,j|k=j}^{(v)}\right)}\right)$$

とかける。（Golfin 2017）

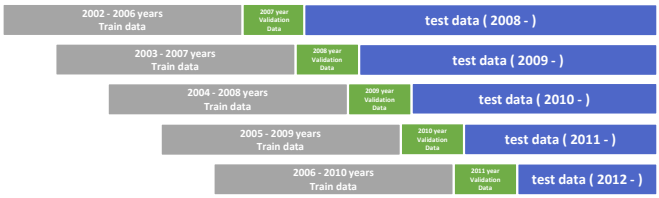
以上の KCM と OPM の考え方を数式を交えて説明しているものが、こちらのスライドとなります。時間の関係上、詳しく数式の中身を追うことはできませんが、基本的に、こちらのスライドの上部ですね。青枠内にありますようなノーテーションと設定のもとで、事故年度 i の最終累計支払/発生保険金は、スライド中段のような数式で書くことができます。括弧の中の左側の項が、実績値、すなわち既に分かっている値、右側の項が予測部に対応する項となっています。

ここまでは、特に KCM・OPM の枠組みを一切入れたものにはなっていませんが、前のスライドでご説明いたしました KCM・OPM の考え方を盛り込んで拡張いたしますと、スライド下段のような式にすることができます。下段の式におけます青色の部分ですね。見にくくて申し訳ございません。そちらの方は、 t 年度以降の将来予測値に対応する部分となります。数式の形から、クレーム請求が終了していないという条件のもとでKCMによる保険金額が計算されるモデルであることが、雰囲気でお分かりいただければと思います。時間の関係上、数式の説明はこれ以上はできないのですけれども、興味のある方は、参照論文等をご紹介できますので、おっしゃっていただければと思います。

B4. データの分割

機械学習・深層学習モデルにおけるハイパーパラメータの決定

- **正則化 GLM における正則化パラメータのように、**
  機械学習・深層学習モデルは各々のモデル固有のパラメータをもつ
  → 「ハイパーパラメータ」と呼ぶ
- **test data 以外のデータを train data と validation data に分割し、**
  train data で fitting を行ったモデルが、
  validation data において精度がでるような、ハイパーパラメータを探索する
- **時系列性のある支払備金の予測モデルのような場合は、**
  例えば、下図のように、年度（CY）で
  test data, train data, validation data を分割する



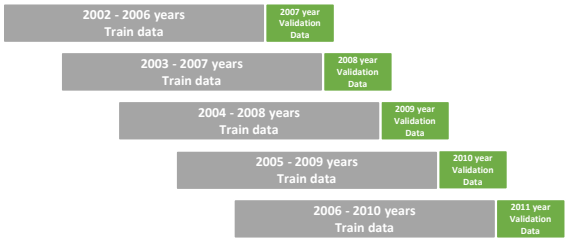
それでは、機械学習・深層学習のモデルの詳細に関して、説明をいたします。最初に、先ほどからvalidationと申し上げておりますが、なぜデータを分割する必要があるのかということに関しまして、ご説明をさせていただきます。このスライドにも記載させていただいておりますとおり、機械学習・深層学習モデルは、ハイパーパラメータというパラメータがございます。そちらを決定するためには、データを分割する必要があります。

聞き慣れない方には、「ハイパーパラメータって何だろう」と思われる方もいるとは思うのですけれども、例えば、正則化GLMの場合、罰則項に対応するGLMの正則化パラメータをどうしても決めなければいけない、どのパラメータがいいかを判断するときに、このような枠組みが必要になるとお考えいただければと思います。正則化GLMのハイパーパラメータの数は、１つや２つなのですけれども、機械学習・深層学習の場合は、これらのハイパーパラメータの数が非常に多くなります。最低でも、３つから４つ程度あります。それらを決定したうえで、改めてモデルのフィッティング、予測を行うことになります。

実際にどのようにやるかということなのですけれども、予測に使用するデータ、これは、先ほどトライアングルで出てきた緑色のセルに該当するようなイメージですね。そちらをtest dataと読んでおりまして、test data以外のデータを、更にtrain data、validation dataと呼ばれるデータに分割いたします。下の図が、それに対応しております。分割のしかたなのですけれども、今回のように時系列性のあるデータの場合は、CYで分割することになります。ハイパーパラメータの決定に必要なものは、実際のところ、train dataとvalidation dataの２つのみになります。test dataは予測する場所ですので。

B4. データの分割

本検討におけるハイパーパラメータの決定



- **5 つの validation data における評価指標（後述）の平均値が ベストとなるようなハイパーパラメータを各モデルにて探索し採用**
- **KCM と OPM は、それぞれ異なるハイパーパラメータを持ちうる**
- **KCM は、RMSE（ $= \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$ ）を、 OPM は、 AUC（受信者感度曲線の下側面積）を、 モデル精度を判断する評価指標として採用**

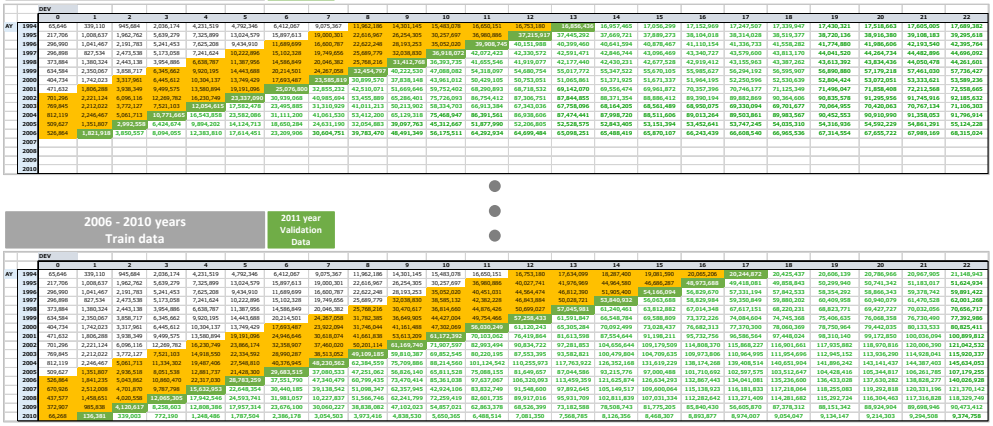
次のページで、実際にどのように分割したかをご紹介したいと思います。本ワーキングパーティーでは、５とおりのデータの分割を行いました。それぞれで train data のモデルのフィッティングを行っています。フィッティングを行いましたモデルの性能は、では、どのように評価するか。それを train data で行ってしまっては、意味がないですね。train data の目的値に合わせに行っていますので。そのため、モデルの性能は validation data で評価いたします。評価した性能の結果は５つ分ありますので、５つの平均値がベストとなるようなハイパーパラメータを、各モデルにおいて探索することになります。

探索の行い方ですけれども、幾つか手法がございます。例えば、一定の範囲内でハイパーパラメータの取りうる値の組み合わせを全点探索する、グリッドサーチというものがありますけれども、非常に計算コストが高いです。ですが、本ワーキングパーティーでは、そちらを採用いたしました。後ほど、各モデルのハイパーパラメータの探索範囲を、各モデルの説明をしながらご紹介いたします。また、KCM と OPM は、異なるハイパーパラメータを持つことを許した形となっております。KCM は回帰モデルですので、精度評価には RMSE を、OPM は分類モデルですので AUC を、モデル精度を判断する評価指標として採用いたしました。

B4. データの分割

データ粒度は、各クレーム毎だが、
データの分割範囲をトライアングルデータ上で示すと・・・

黄色のセル は train data
緑色のセル は validation data
緑文字のセル は test data



トライアングルデータ上で、これまでお話しいたしましたデータの分割のしかたを見てみます。そうしますと、先ほどのスライドの最初の分割パターンですと、このスライドの上図のような形になります。黄色のセルが train data、緑色のセルが validation data ということで、1年分なのですね、validation data は。そのような形でデータの分割をすることになります。

それでは、機械学習・深層学習モデルの説明に、いよいよ入っていきたいと思います。一旦、各種アルゴリズムの説明をひととおりいたします。お時間の関係がありますので、比較的駆け足になってしまうと思います。

B5. 機械学習・深層学習モデル

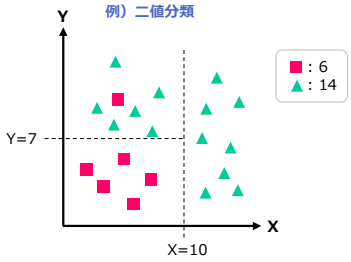
機械学習・深層学習のモデルトレンド

1. 決定木系のアルゴリズム

- Random Forest
- Boosting Machine

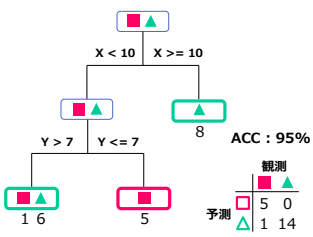
決定木

例）二値分類



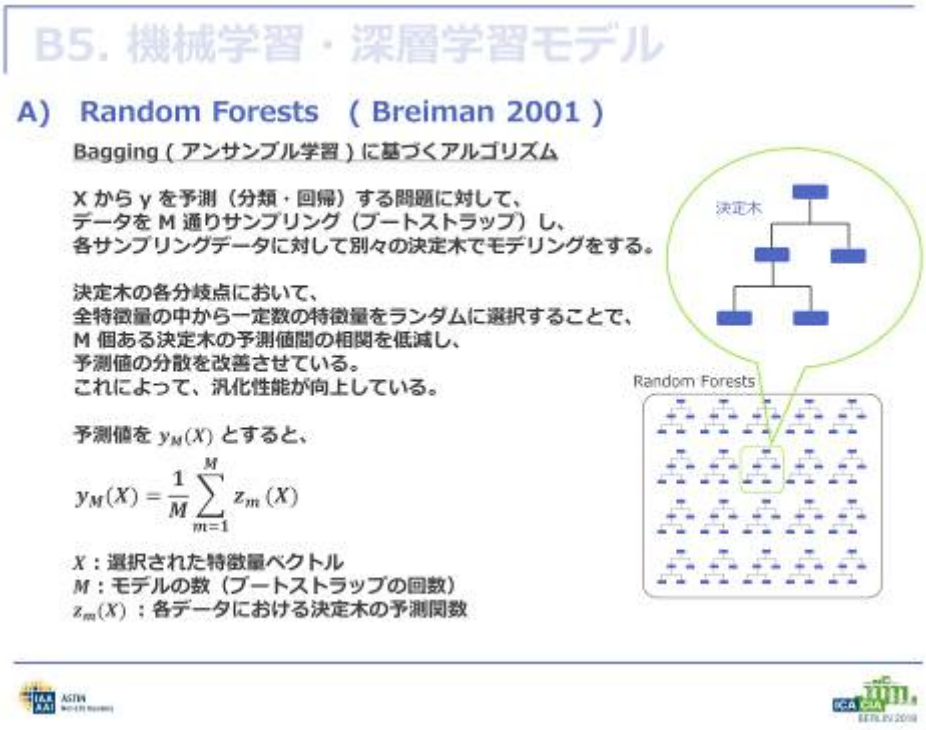
2. ニューラルネットワーク系のアルゴリズム

- DNN
- CNN
- RNN / LSTM / GRU



本ワーキングパーティーで使用しました機械学習・深層学習のモデルは、大きく分けまして、決定木系の機械学習アルゴリズムと、ニューラルネットワークという深層学習系のアルゴリズムとなります。昨今のデータサイエンスにおけるモデルのトレンド、はやりなのですけれども、決定木系のアルゴリズムが非常にはやっております。例えば、Random Forest や Boosting Machine など、そのあたりが非常にはやっております。ニューラルネットワーク系のアルゴリズムですと、ディープニューラルネットワーク、畳み込みニューラルネットワーク、再帰型ニューラルネットワークなどが、よく利用されております。本ワーキングパーティーで実際に使用しましたものは、こちらのアルゴリズムとなっています。赤枠で囲った部分ですね。後ほどニューラルネットワークに関しましては、ベーシックな説明を簡単にさせていただきますので、ここでは決定木系のアルゴリズムの基となる決定木に関して、右図で簡単にご説明をさせていただきます。

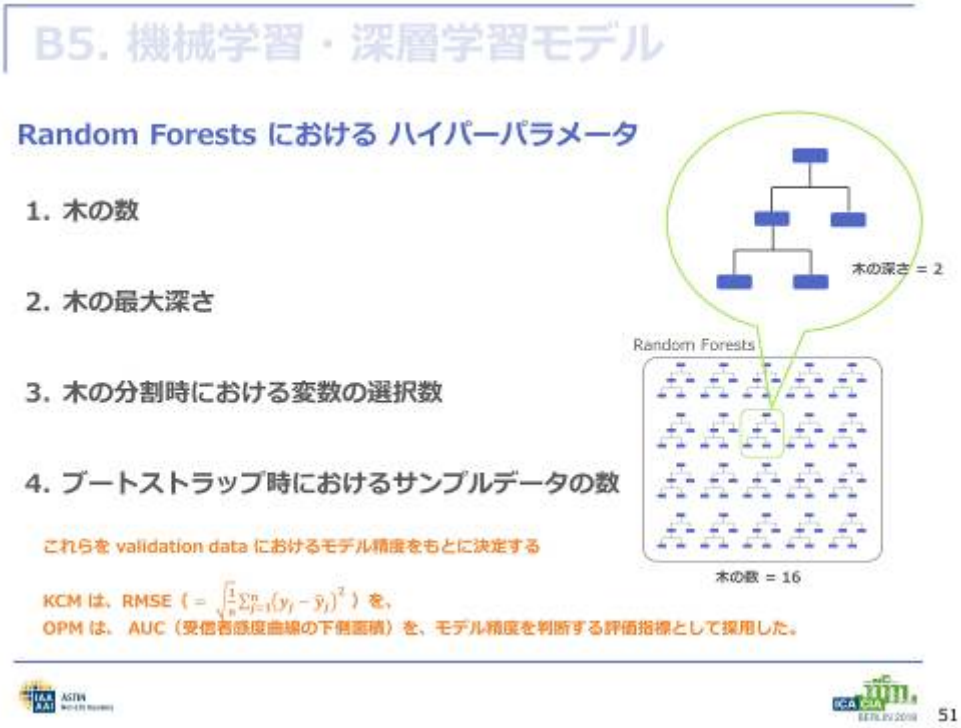
ここで例として出しておりますのは、三角と四角の２つのクラスが、X と Y という値で決まっているようなデータを、線形のアルゴリズムでうまく分類モデルを作って分割しようといたしますと、難しいことがお分かりいただけるかと思います。一方で、決定木系ですと、右下の図にありますように、最初に X が 10 以上か、それより小さいか、次に Y が７より大きいか、それ以下かという分け方をいたしますと、少し恣意的な例ではありますが、正解率 95％ということで、うまく三角と四角の２つのクラスを分類できることになります。つまり、従来の線形モデルではうまく分類できなかったものが、決定木系の非線形アルゴリズムでは非常にきれいに分類できるということで、決定木系アルゴリズムのパワフルなところが、かいま見えるかと思います。



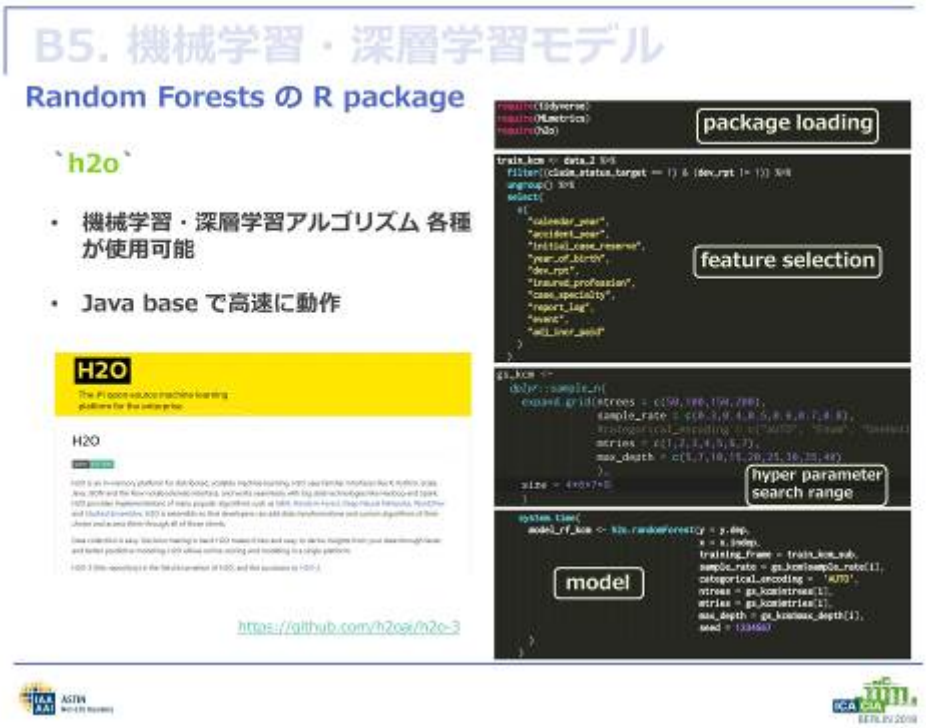
それでは、最初に、Random Forest に関して説明をいたします。こちらは、決定木を幾つも使用したものでして、「Bagging」と呼ばれるアンサンブル学習に基づいたアルゴリズムとなります。アンサンブル学習という言葉をお聞きになったことがあるかと思いますけれども、アンサンブル学習には Bagging 以外には、Boosting というものもございます。

特徴としては ２点ほどございます。１つ目はモデルのアルゴリズムに関してです。この手法ではデータ

をM通り、ブートストラップでサンプリングします。各サンプリングデータに対して、それぞれの決定木でフィッティングする。そして、予測を行うと。M通り予測を行うので、木がM個できるわけですね。２つ目のポイントですけれども、木を分割するときに、全特徴量を使うのではなく、一定数の特徴量をランダムに選択するという点がポイントとなります。なぜこのようなことをやるのか、理論は今日は追いかけないのですけれども、これらを行うことで予測値の分散が低減されまして、モデルの期待損失を改善させることができます。下式にありますように、予測値は、全決定木の平均値となります。



データの分割のしかたでお話しいたしました、機械学習特有のハイパーパラメータですけれども、Random Forest の場合、重要なものとして４つ挙げられます。１つ目が木の数です。２つ目は、それぞれの木が許される最大の深さ。例えば、右図ですと「深さ＝２」と書いています。３つ目が、先ほどお話しいたしました、木の分割時における変数の選択数です。最後は、ブートストラップ時におけるサンプリングデータの数。全体のどれくらいをサンプリングするのかというところになります。基本的には、一番目の木の数は多ければ多いほどいいのですが、計算時間との兼ね合いもございますので、十分に精度が出て、収束するような木の数を選択することになります。



使用いたしましたRのパッケージですけれども、h2oのRandom Forestパッケージを使用いたしました。h2oパッケージは、Random Forestだけに限らない非常に包括的なパッケージでして、他にもいろいろなモデルが使用可能です。また、バックエンドがJava baseですので、かなり高速にR上でも動作します。右図がコードの一部を示したものになりますが、上から、パッケージの読み込み、使用している特徴量の一覧となります。そして、その下が探索したハイパーパラメータのレンジです。グリッドサーチをするためのグリッドを表しています。最後がモデル構築部となります。これが一連の流れとなります。結果に関しましては、このスライドの部分では説明いたしません。最後に他のモデルとあわせてご説明させていただきます。

B5. 機械学習・深層学習モデル

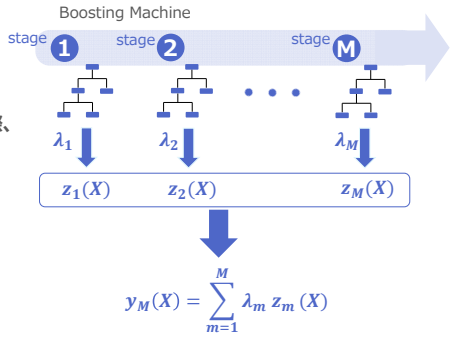
C) Gradient Boosting Machine (Friedman 2001)
(勾配 boosting)

Boosting に基づくアルゴリズム

X から y を予測（分類・回帰）する問題に対して、決定木を M 個分、逐次的に学習させる。

例えば、i （ = 2, 3, …, M ）番目の木を学習させる際、i-1 番目の木までの予測結果の誤差を補正するように、i 番目の木の構造を決定する。
「勾配 boosting」の場合は、誤差関数の負の勾配方向に沿うように、決定木の構造を fit させる。
予測値を $y_M(X)$ とすると、

$$y_M(X) = \sum_{m=1}^{M} \lambda_m z_m(X)$$

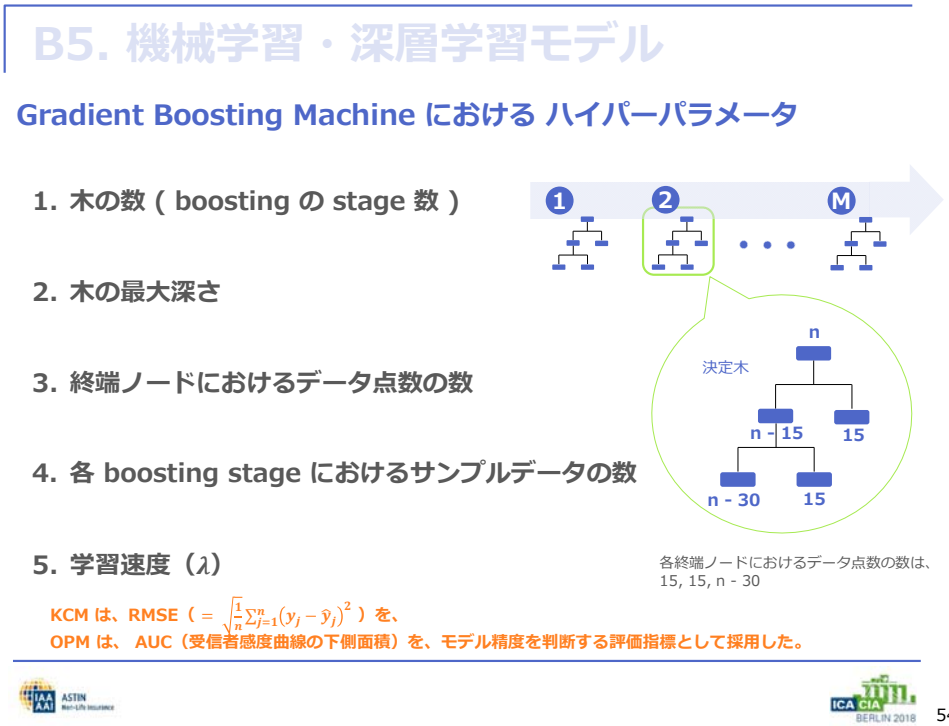


X : 選択された特徴量ベクトル
M : モデル（木）の数 （ boosting の stage 数 ）
$z_m(X)$: 各 boosting の stage （ 1 ~ M ） における決定木の予測関数
$\lambda_m(X)$: 各 boosting の stage （ 1 ~ M ） における決定木の重み （ 学習速度 ）

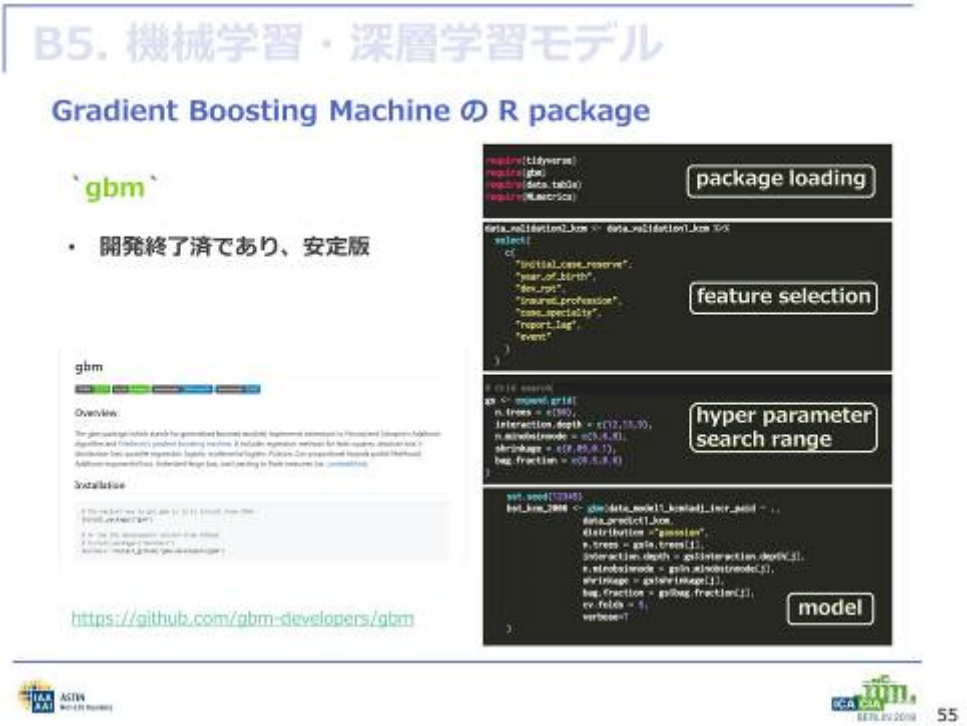
続きまして、こちらはデータサイエンスのアルゴリズムの中でも大本命なのですけれども、Gradient Boosting Machine というアルゴリズムに関して、簡単にご説明させていただきます。Random Forest と対称的なアルゴリズムとして挙げられるものです。Boosting という概念に基づいております。Random Forest は、木の数分、並列的に学習を行って、最後に平均を取るということを行ったのに対しまして、Boosting アルゴリズムは、逐次的に、直列に学習を行います。

どのようなことかというと、右図をごらんいただければイメージがつくかと思いますが、ステージ１、ステージ２、ステージMとあります。M個分の木を学習させることはさせるのですが、最初の１個目の木の学習が終わったら、次の２個目に行くということで、シーケンシャルに続いているわけですね。では、どのように学習するのかというお話になってくるのですけれども、例えば、ｉ番目の決定木の学習を行う場合、ｉ－１番目までの木の全てで構成される予測値、予測結果の誤差を、補正するように学習を行うことになります。ですから、前の木までの出力が、非常に重要なわけですね。

使っております Gradient Boosting Machine は、「Gradient」とありますので、勾配なのですけれども、勾配 boosting と呼ばれます。どのようなことかといいますと、勾配 Boosting は、i 番目の木の出力を決める際に、誤差関数の負の勾配方向に沿うように、i 番目の決定木の構造を決めることになります。スライドの左下にあります数式が、予測値を表すものとなります。RnadomForest のように平均値にはなっておらず、各木の出力がある決められた重みで、加重平均のような形で計算されていることが分かると思います。こちらの重みの決め方までは、本日は説明いたしません。

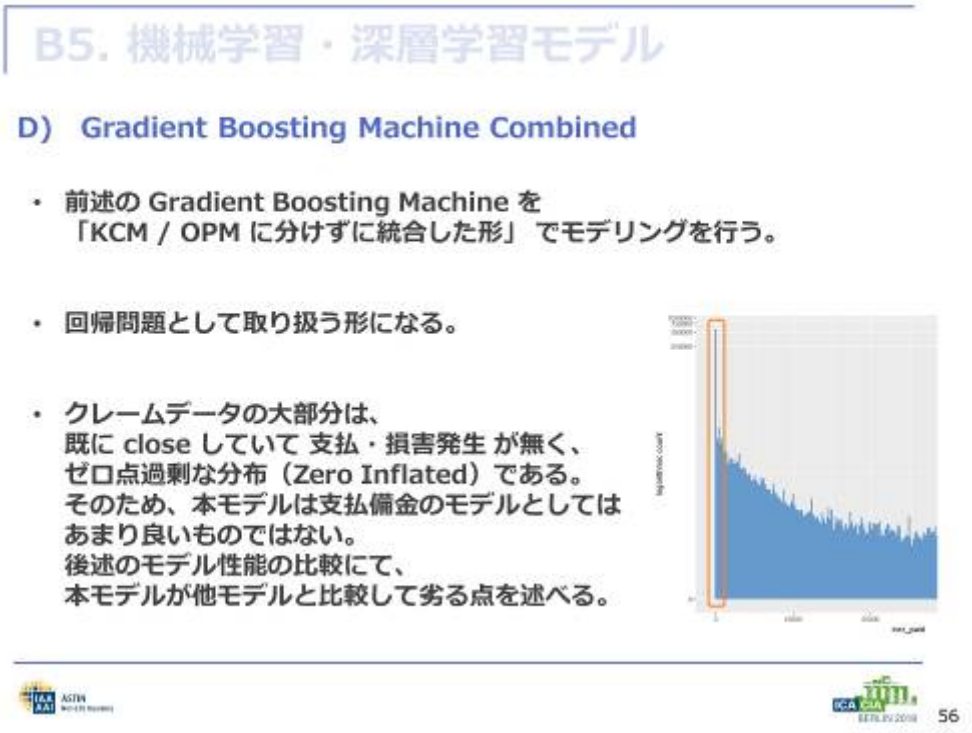


勾配 boosting のハイパーパラメータですけれども、重要なものを、スライドに５つほど挙げさせていただきました。どれも重要なのですが、特に木の深さは重要です。他には、木の終端ノードにおけるデータの点数など、このあたりも重要になります。



そして、勾配 boosting の R のパッケージですけれども、gbm というパッケージを、本ワーキングパーティーでは使用いたしました。このパッケージは、かなり前からあるもので、かつ、既に開発は終了しています。ですから、今は使う人はあまりいないのかもしれませんが、本ワーキングパーティーでは採用いたしております。右図が、R のコードの一例です。先ほどと同じような一連の流れを図示したものとなってい

ます。



次に、こちらの Gradient Boosting Machine の Combined というものがあるのですけれども、こちらは何かと申しますと、これまでのモデルでは KCM と OPM はそれぞれ別々にモデリングいたしますが、別々にモデリングせずに、統合した形でモデリングを行うものになります。ただ、結果から申しますと、あまり良いモデルではありません。なぜかといいますと、右下の図にありますように、支払保険金や発生保険金は、ゼロ点過剰な分布なわけですね。最近の機械学習のモデルといえども、なかなかこのような分布を表現することは難しいということで、後ほど他のモデルと合わせてご説明いたしますが、あまり良い結果には至りませんでした。

B5. 機械学習・深層学習モデル

E) Boosted Tweedie Compound Poisson Model (Yang et al. 2016)

- Tweedie 分布と GBM を組み合わせたモデル
- Tweedie 分布は、

$$Var(y) = \frac{\phi}{\omega} \cdot E[y]^p$$
$$y \sim Tweedie(\mu, \phi/\omega, p)$$

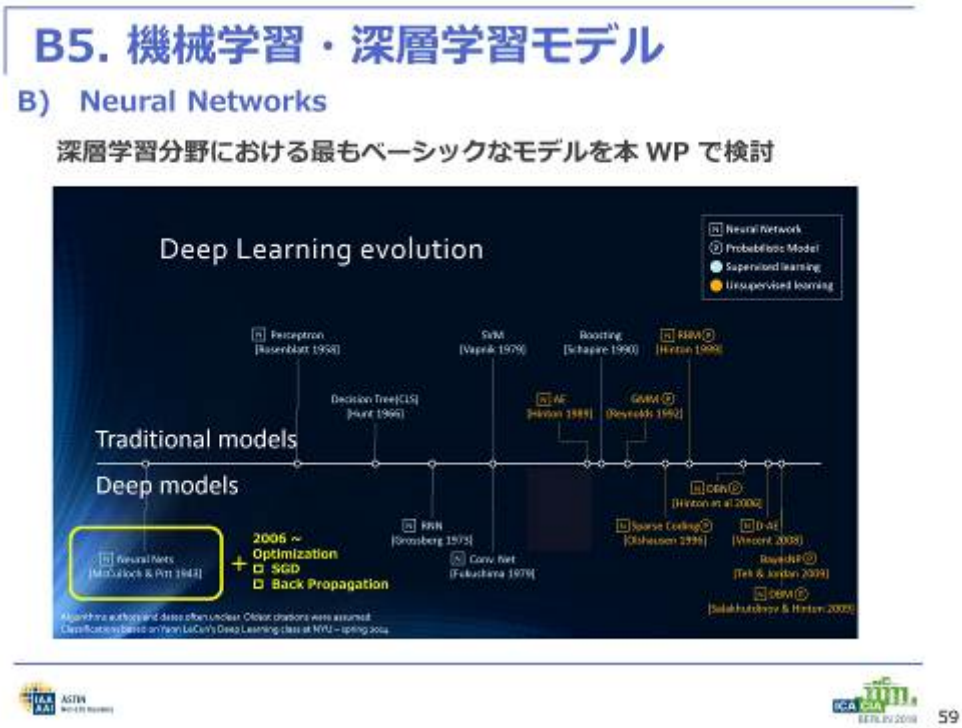
と表せる。ここで、ϕ は分散パラメータ、ωは エクスポージャを表す。
p は乗法パラメータであり、
$p = 0$　　正規分布
$p = 1$　　ポアソン分布
$p = 2$　　ガンマ分布
$1 < p < 2$　　複合ポアソンガンマ分布　　となる。

- 本 WP では、
KCM は $p = 2$ すなわち ガンマ分布 を、
OPM は $p = 1.001$ すなわち 「ほぼ」ポアソン分布 を採用した。
- その他の設定は、前述の GBM と同様。

ということで、残り２個ほどですね、モデルは。そのうちの１つが、Boosted Tweedie Compound Poisson Model となります。これは、Tweedie 分布をお聞きになった方も中にはいらっしゃるかとは思いますが、勾配 boosting と Tweedie 分布を組み合わせたモデルとなります。Tweedie 分布は、指数分布族の特別な形であるということはご存じの方も多いと思うのですが、本スライドの中程で示されるような特徴を持っています。そちらのモデルをどのように使ったかといいますと、GLM の枠組みでリンク関数は当然使うと思うのですが、リンク関数の線形表現されていた部分に、この木の回帰モデルの出力を置き換えるという形になっています。これはなかなかいいモデルでして、機械学習と従来のパラメトリックなモデルのいいとこ取りのようなイメージを私は持っています。

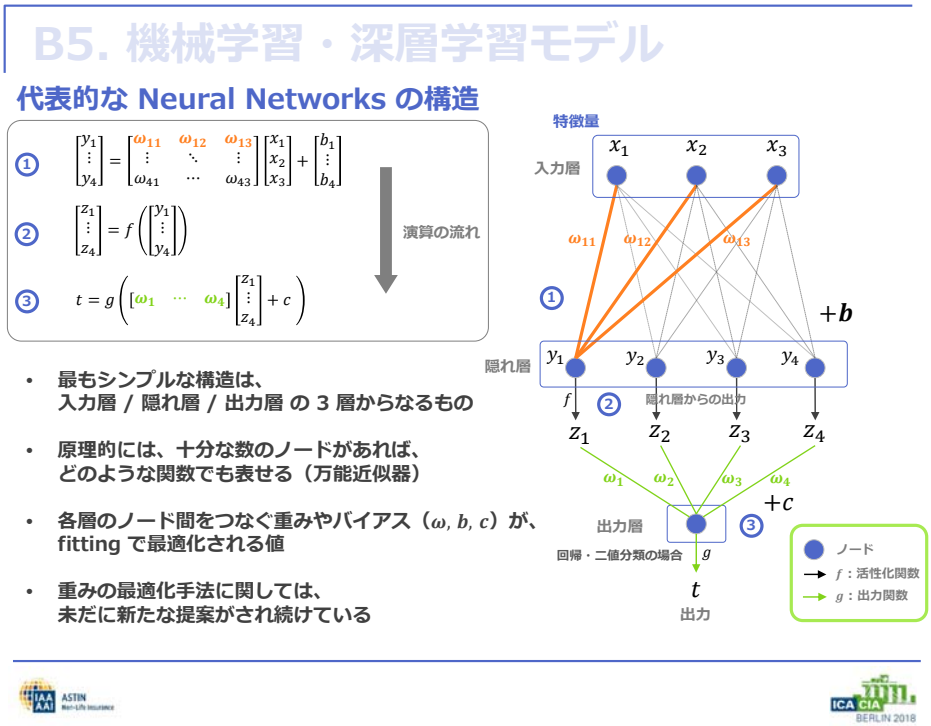
本ワーキングパーティーでは、乗法パラメータ、p ですね。中程に「p」と書いてありますけれども、そちらも、正規分布からポアソン分布まで、いろいろな表現は当然できるのですけれども、KCM の場合は p＝２、すなわちガンマ分布としています。OPM は p＝１が使えないので、1.001 ということで、ほぼポアソン分布であるような分布を採用いたしました。

こちらが、使用いたしました R のパッケージとなります。あまりメジャーなパッケージではないのですけれども、TDboost というパッケージを利用いたしました。近年、データサイエンティストがよく使用するパッケージで xgboost というものがあるのですけれども、そちらでもこのような Tweedie 分布は指定できますので、あえてこれを使う必要はないかなと思う面もございます。R のコードの一部に関しては、右図のようになります。



最後のモデルといたしまして、ニューラルネットワークのモデルに関して、簡単にご説明をさせていただきます。こちらが、ニューラルネットワークの技術を時系列で表したものになるのですけれども、少し

見にくいですね。すみません。本日ご紹介しますニューラルネットワークは、最もベーシックなモデルとなります。最近ですと、画像処理、自然言語処理の分野で、畳み込みニューラルネットワークや再帰型ニューラルネットワークなどの研究開発が盛んなのですが、今から遡ること 1943 年頃に、アニメーションで示している部分ですが、既にニューラルネットワークの基礎理論が確立されています。ただ、計算機の制約や、ネットワークの学習、具体的には中で最適化される重みを求めるということなのですけれども、その辺がうまくできないということで、問題を抱えておりました。ただ、2006 年頃から、確率的勾配降下法や、Back Propagation といった技術の開発とともに実用化のめどが立ったということで、昨今、非常に話題となっている分野になります。



こちらが、代表的なニューラルネットワークの構造となります。基本的には、３層のニューラルネットワーク構造が代表的なものとなります。入力層と隠れ層、出力層という３つの層から成り立っておりまして、入力層は、説明変数、特徴量に対応する層となります。入力された特徴量が出力層までどのように伝播していくかを表したものが、左上の数式となっております。興味のある方は、追いかけていただければと思います。

ポイントといたしましては、３点ございます。隠れ層からの出力は、活性化関数と呼ばれる非線形な関数で変換を受けるという点が１つ目です。２つ目が、訓練データで、各層のノード間をつなぐ重み、w と表記していますけれども、あとは、バイアス b ですね、それらが最適化されて求められるという点になります。そして、３つ目ですけれども、出力層からの出力は、出力関数によって、解くべき問題に適した出力構造に変換されるというものになります。例えば回帰問題でしたら、それに対応したスカラー値を表す単一ノードに、３つ以上のクラス分類でしたら、出力ノードは出力ベクトルに対応したクラス数分のものとなります。GLM は、指数分布族などの限られた分布を表現いたしますが、一方、このような構造を持つニューラルネットワークは、十分な数のノードがあれば、どのような関数でも表現が可能です。そのため、ニューラルネットワークは「万能近似器」とも呼ばれます。

B5. 機械学習・深層学習モデル

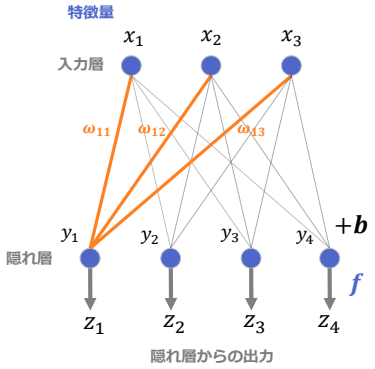
活性化関数

① $\begin{bmatrix} y_1 \\ \vdots \\ y_4 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \vdots & \ddots & \vdots \\ \omega_{41} & \cdots & \omega_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_4 \end{bmatrix}$

隠れ層への入力　重み行列　特徴量　バイアス

② $\begin{bmatrix} z_1 \\ \vdots \\ z_4 \end{bmatrix} = f\left(\begin{bmatrix} y_1 \\ \vdots \\ y_4 \end{bmatrix} \right)$

隠れ層からの出力　f：活性化関数



- 活性化関数は、隠れ層からの出力を演算する場合は、非線形な関数を使用する。
  非線形な関数は、本 WP で使用したもの以外にも多数のものが提案されている。
- 出力関数の場合は、回帰モデルであれば、線形である恒等関数を使用し、
  分類モデルであれば、sigmoid や softmax などを使用する。

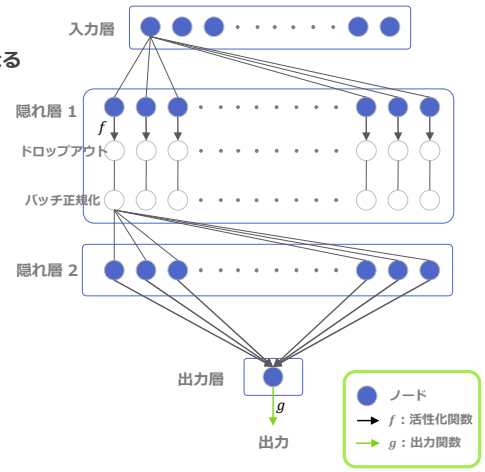
61

次に、入力層と隠れ層の演算をもう少しだけ詳しく見てみたいと思います。特徴量が x_1 から x_3 とありますが、入力層の各ノードにそれぞれが対応いたします。その場合、左上の数式①のように、入力層と隠れ層のノード間をつなぐ重みによって、隠れ層の各ノードを表す値へと変換を受けます。この重みとバイアスが、最適化されるべき値です。そして、数式②のように、非線形な活性関数によって、隠れ層の出力となるような変換を受けます。仮に活性化関数が非線形ではなく線形であった場合、計算してみると分かるのですけれども、ネットワーク全体として、ただの重回帰モデルとなってしまいます。ですから、活性化関数は、非線形であることが重要です。

B5. 機械学習・深層学習モデル

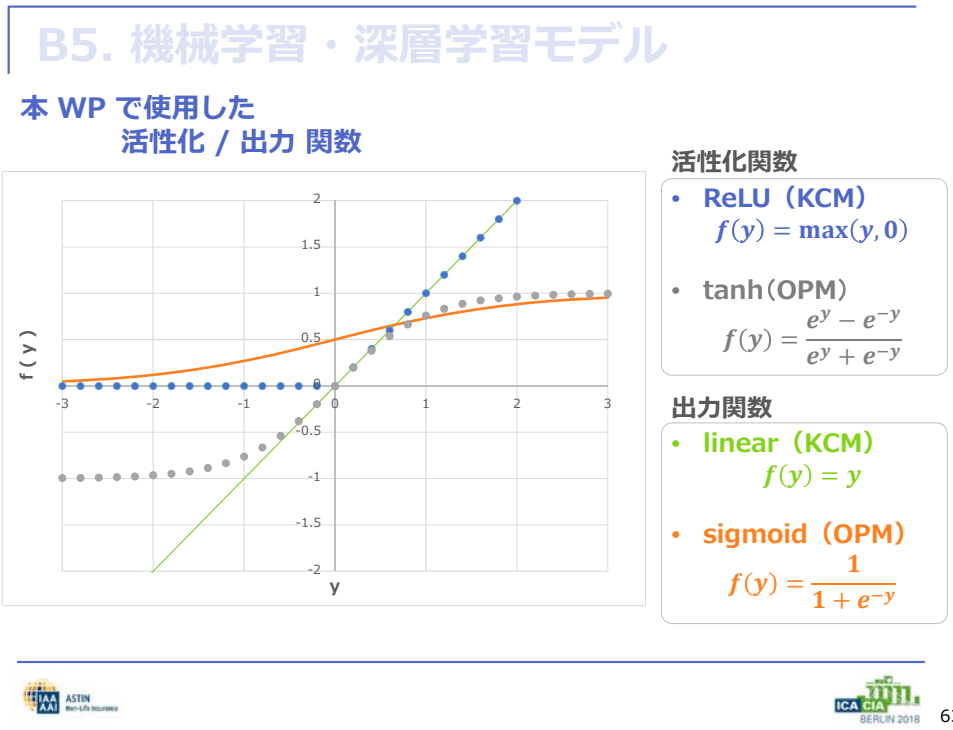
本 WP で使用した Neural Networks の構造

- 本 WP では、
  入力層、隠れ層 1・2、出力層 の 4層からなる
  ネットワークを、KCM 及び OPM に適用
- 隠れ層1 のノード数は、500
  隠れ層2 のノード数は、250
- KCM
  隠れ層の活性化関数は ReLU
  出力関数は linear
  OPM
  隠れ層の活性化関数は tanh
  出力関数は sigmoid
- ドロップアウト層のドロップ率は 10%
- バッチ正規化の位置が通常と異なる



62

それでは、実際に使ったネットワークの構造を紹介いたします。先ほどの例とは異なりまして、隠れ層は１つ増えております。全部で４つの層からなるネットワークとなります。次のページ以降で、こちらに関しまして、説明をさせていただきたいと思います。主に活性化関数、ドロップアウト、バッチ正規化、以上の３つの技術に関して説明させていただきます。



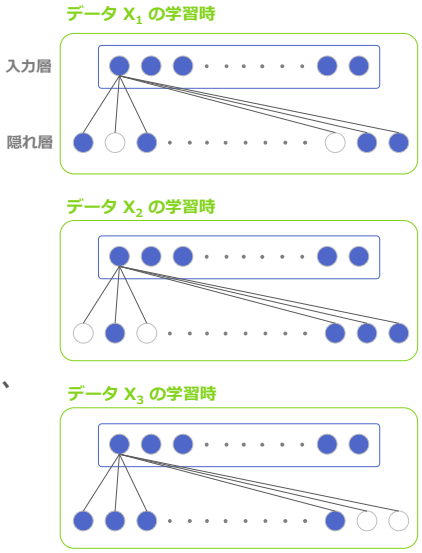
最初に活性化関数なのですけれども、KCM の場合は、隠れ層の活性化関数は ReLU 関数、これは「レル」と読みます。そして、出力関数は linear、ただの恒等関数ですね。そちらを使用いたしました。OPM の場合は、隠れ層の活性関数は tanh（タンジェントハイパボリック）、出力関数は sigmoid。これは、二値分類だからですね。そちらを使用いたしました。活性化関数には他にもいろいろな種類がありますが、ハイパーパラメータと同様に、validation におけるモデル精度を判断して選択することになります。

B5. 機械学習・深層学習モデル

ドロップアウト

- 隠れ層のノードのうち、
  いくつかを使用せずに学習を行う。
- 一般的には、
  ドロップ率は 0.5 付近を使用するが、
  validation を通して最適化する。
- ドロップアウトは、データ毎に、
  いくつかの特徴量を使用せず、
  残りを使用するというイメージに近い。
- Random Forest と同様に、
  使用する特徴量をランダムで選択することで、
  高い汎化性能をもたせていると、
  みなすことができる。

**ドロップアウトは、
擬似的に、
アンサンブル学習を行っている。**

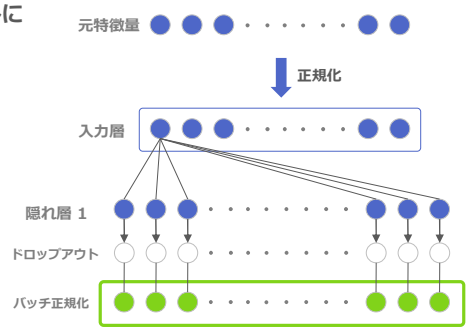


次に、ドロップアウトに関して説明いたします。こちらは、比較的最近出てきました技術ではあるのですけれども、学習の際に、隠れ層のノードをある一定の割合で使用しないで、モデルの汎化性能を上げる手法になります。汎化性能と言っているのは、train data のモデルに対して過剰に適合しないようにする。つまり、未知のデータが入ってきたときに、未知のデータに対しても一定の精度をきちんと出せるようにするということになります。使用しない割合は、これも結局 validation ロジックを通して適した値を見つけることになりますが、大体 50%前後が目安となります。Random Forest の説明を思い出していただけるといいかと思うのですけれども、ドロップアウトは結局何をやっているかというと、一定の割合でノードを使わないようにしているので、ノードに対応したメタな特徴量が伝播していくのを一定の割合で塞いでいるわけですね。ですから、Random Forest でありましたように、変数を一部選択するような機能があると思います。ということで、汎化性能の向上に寄与するというイメージがつくと思います。

B5. 機械学習・深層学習モデル

バッチ正規化

- 深層学習は重みを最適化で求めることになるが、
  最適化のアルゴリズムの関係上、
  重みの初期値や入力する特徴量のスケールに
  最適化の良し悪しが大きく影響を受ける。
- 特徴量のスケールを均一にするため、
  入力する特徴量を必ず正規化するが、
  伝播するにつれて、
  各ノードにおける値のスケールは
  大きく異なってしまう。
- そこで、主に隠れ層間に
  「バッチ正規化」と呼ばれる処理を
  挿入し、スケールを揃える。
- 通常は、
  活性化関数の直前にバッチ正規化をいれるが、
  本 WP では、ドロップアウトの直後に使用している。



最後に、バッチ正規化に関して説明をさせていただきます。学習を通して最適化されるのは、ノード間の重みとバイアスでした。使用している最適化のアルゴリズムに依存はいたしますけれども、重みの初期値や特徴量のスケールに最適化の良し悪しが大きく依存されてしまいます。そのため、入力する特徴量は、基本的には正規化をしたりして均一な値にスケーリングするのですけれども、入力層から出力層に向かって隠れ層をどんどん進むに従いまして、重み係数をいろいろと掛けたあとの各ノードの値は、ノード間でばらついてしまいます。非常に大きな値もあれば、小さな値もありますので、ニューラルネットワークでは、バッチ正規化と呼ばれる処理を、一般的には活性化関数の直前に入れることになります。本ワーキングパーティーではドロップアウトの直後に入れておりまして、この点は、実は少し「どうなのかな」ということはあるのですけれども、いずれにいたしましても、バッチ正規化を入れていることは確かです。

B5. 機械学習・深層学習モデル

バッチ正規化

- 実際の学習時では、
  データ 1 つ毎に重みを最適化するわけではなく、
  一定の数のデータに対して重みを最適化する。
  これを「バッチ（ミニバッチ）学習」と呼ぶ。

  まず、m 個のデータからなるミニバッチ $\beta = \{x_1, x_2, x_3, \cdots, x_m\}$ があった場合、
  ミニバッチの平均 μ_β および分散 σ_β^2 は、

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2$$

  と記述できる。
  これに対して、バッチ正規化は、ミニバッチ内の各データ x_i を下記のように変換する。

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta$$

  ここで、γ と β は、バッチ正規化にともなうモデルパラメータであり、最適化の対象である。

次のページで、バッチ正規化のロジックを、もう少しだけ詳しく見てみたいと思います。そもそも「バッチ」という言葉は何なのかということですが、ニューラルネットワークの学習時に、データを１つずつ使用してネットワークの学習をするのではなくて、一定数のデータをまとめて学習させる「バッチ学習」という言葉から来ています。基本的には、ニューラルネットワークのモデルを学習させる時は、バッチ学習をすることになります。本スライドの数式の説明では、データがm個あったとして、バッチ正規化の説明を書かせていただいております。これは非常にシンプルな、標準化のようなものですね。ここで行っていることは、m個のサンプルに対して、その平均・分散を計算して標準化するということになります。

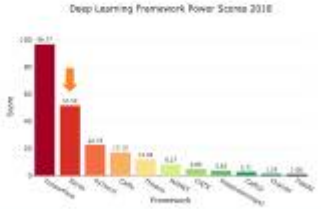
B5. 機械学習・深層学習モデル

Neural Networks の R package

`Keras` (TensorFlow backend)

TensorFlow Keras

- TensorFlow は、
  google が開発した深層学習ライブラリ
- TensorFlow のままだと
  コードが書きにくいので、
  それを助ける wrapper package が、
  Keras




```
require(tidyverse)
require(MLmetrics)
require(keras)
require(tensorflow)
```
package loading

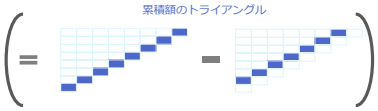
```
train_kcm <- data_2 %>%
  filter((claim_status_target == 1) & (dev_rpt != (1-min_dev))) %>%
  ungroup() %>%
  select(
    c(
      "dev",
      "claim_number",
      "calendar_year",
      "accident_year",
      "initial_case_reserve",
      "age_group",
      "dev_rpt",
      "insured_profession",
      "case_specialty",
      #"incident_grouping",
      "report_lag",
      #"event",
      "adj_incr_paid"
    )
  )
```
feature selection

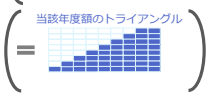
```
# KCM NN MODEL
model_kcm_orig <- keras_model_sequential()
model_kcm_orig %>%
  # layer 1
  layer_dense(units = 500,
              input_shape = c(length(x_indep_kcm)),
              activation = 'relu') %>%
  layer_batch_normalization() %>%
  # layer 2
  layer_dense(units = 250,
              activation = 'relu') %>%
  layer_batch_normalization() %>%
  # output
  layer_dense(units = 1, activation = 'linear')

summary(model_kcm_orig)
```
model

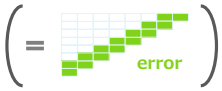
さて、本ワーキングパーティーで使用しましたニューラルネットワークの R パッケージですけれども、プログラムコード自体は、Keras というパッケージを使用してネットワーク構造を記述しております。バックエンドは、google が開発した TensorFlow を使用しております。元々TensorFlow でもネットワークのプログラムを書くことはできるのですが、少し書きにくいものですから、wrapper package である Keras を使って、モデリング、プログラムコードを書くということが、割と現状としては多いかと思います。以上で、伝統的な従来手法と機械学習・深層学習的手法モデルのご紹介を終えたいと思います。

伝統的手法と機械学習・深層学習の計算結果比較

- **2006, 2011, 2014 および 2016 年を評価年度とした、バックテストアプローチにて、各モデルの予測値の比較を行う（その時点まで観察できている情報で、どれだけ将来を予測できているかを検証）**
- **評価指標は、以下の 3 つ**

1. 年間（支払 / 発生）保険金 

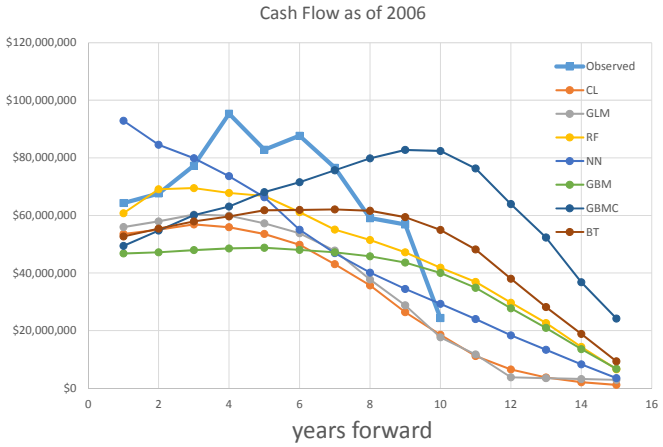
2. 将来（支払 / 発生）保険金 

3. トライアングルにおける各セルの誤差の RMSE（ 誤差 = 予測値 - 観測値 ） 

では、最後に、各モデルの検討結果の比較に移りたいと思います。モデルの精度の良し悪しを、どのように評価したかということなのですが、2006 年、2011 年、2014 年、2016 年の４つの年度を評価年度としたバックテストアプローチに基づき、行いました。評価指標としては、年間の支払/発生保険金の誤差を見るものと、将来の支払保険金・発生保険金の誤差。そして、支払保険金・発生保険金をトライアングルデータの形にした場合の、各セルの誤差の RMSE を取ったものということで、３種類で見ているのですけれども、時間の都合上、こちらの方だけのご紹介にとどめさせていただきたいと思います。

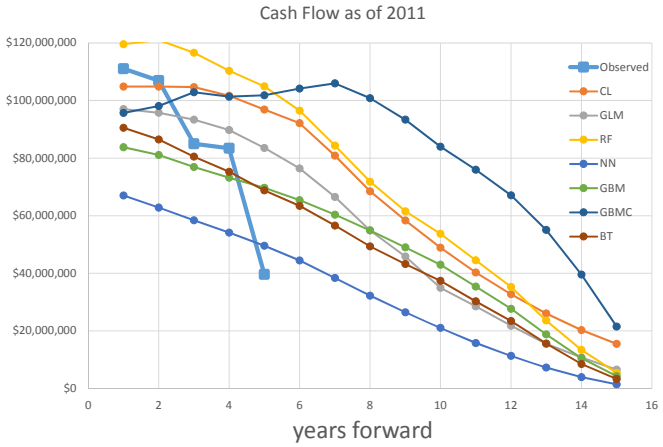
伝統的手法と機械学習・深層学習の計算結果比較

年間支払保険金



こちらが、2006 年を評価年度といたしました、年間支払保険金の実績値と、それに対する各モデルの予測値をプロットしたものとなります。2006 年度が評価年度ですので、1 年後は 2007 年度の支払保険金の値を、2 年後は 2008 年度の支払保険金の値を指すことになります。Gradient Boosting Machine Combined の予測値が、他のモデルと異なっていることが読み取れると思います。また、一方で RMSE の観点では、Random Forest と Boosted Tweedie が、比較的良好な結果を示しています。

伝統的手法と機械学習・深層学習の計算結果比較



次に、こちらが同様に 2011 年度のものとなります。2006 年度と同様に、Gradient Boosting Machine Combined が、少し変なカーブを描いているということになります。

伝統的手法と機械学習・深層学習の計算結果比較

**GBMC の年間支払額が大きく変動しており、
ゼロ点過剰な分布である支払保険金に対して、
良いモデルでないことが伺える。**

71

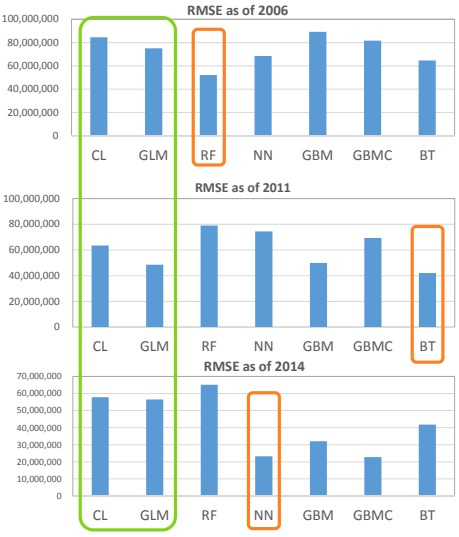
こちらは、評価年度が 2014 年度と 2016 年度のものになりますが、2016 年度のものは、予測値に該当する観察データがございませんので、観察データがプロット上に存在いたしません。これらの２つの評価年度ですと、Gradient Boosting Machine Combined model の年間支払保険金が、プロットに収まらないくらい大きく変動してしまっているということで、ゼロ点過剰な分布に対しては、良いモデルではないということが示唆されます。

伝統的手法と機械学習・深層学習の計算結果比較

**評価年度 2006, 2011, 2014年における
各モデルの RMSE を比較**

$$RMSE_m = \sqrt{\sum_{j=m+1}^{2016} \left(\hat{I}_j^m - I_j^m\right)^2} \quad m \in \{2006, 2011, 2014\}$$

- **総合的にみると、年間支払保険金に関しては、Boosted Tweedie が優れている。**
- **GBMC の 2014年における RMSE が低いが、2016年以降の予測値が、前述の通り不安定である。**

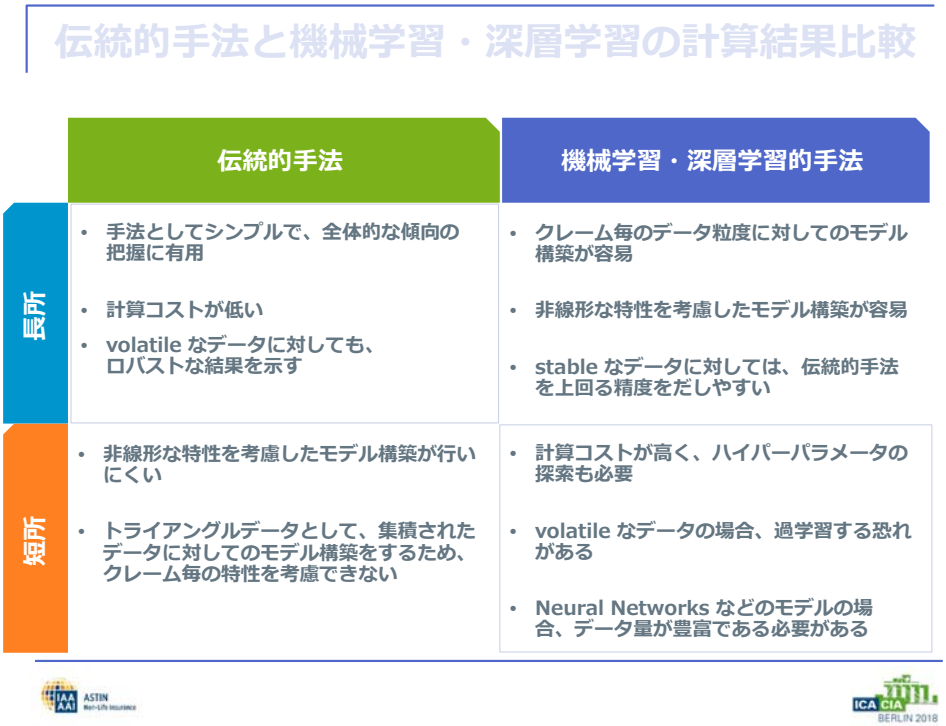


72

こちらが、観察データがございます評価年度の 2006 年、2011 年、2014 年における、各モデルの RMSE を比較したものとなります。右図において緑色で囲みましたものが伝統的な従来的手法で、オレンジ色で囲

みましたものが、各評価年度で最も優れていたモデルとなります。全年度の RMSE を取る形ですと、Boosted Tweedie が、かなり前のページであれなのですけれども、最も優れているという結果になります。

それでは、最後に、全評価指標を支払保険金・発生保険金に関しましてまとめたものを、ご覧いただきます。こちらが、ASTIN WP 2018 のレポートからそのまま表を取ってきたものになるのですが、各評価指標や、発生保険金もしくは支払保険金に関してかという観点で、ベスト、ワーストのモデルが大きく異なります。アニメーションで明滅している部分が、先ほどお見せしたものになります。全体的な傾向としましては、支払保険金ではニューラルネットワークが、発生保険金では勾配 boosting が良い傾向にあります。ただ、一方で、伝統的手法でありますチェインラダーなどは、どの評価指標、評価年度でも、安定した傾向にあるということが言えると思います。



以上を踏まえますと、このような比較結果となります。伝統的手法は、シンプルで計算コストが低い点が、まず挙げられます。また、今回の発生保険金のようなボラティリティーが高いデータでも、安定して推定を行える傾向がある点が挙げられます。一方で機械学習・深層学習的手法は、クレームごとのデータの粒度に対するモデル構築が容易な点。また、非線形性を考慮したモデリングもできる点。あとは、今回の支払保険金のような、比較的データが安定している場合ですね。そちらの場合は、従来手法を上回るような精度を示すモデルが多いという点が挙げられます。１点注意なのですけれども、ニューラルネットワークのようなモデルの場合は、データ点数が十分にないと、なかなかうまく学習できなかったりという欠点がありますので、そこは注意をする必要があると思います。

7. 今後の展望

1. 予測すべき対象と評価指標によって、最良のモデルは異なっていた。

   No Free Lunch Theorem :
   there is no model that is always better than all other models,
   one should try them all and then understand
   which one perform better on the underlying dataset.

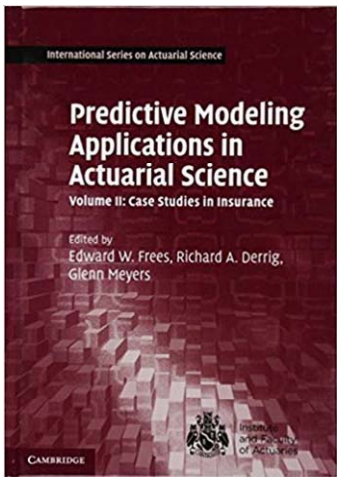
   機械学習・深層学習のモデルの予測値を組み合わせることで、
   よりロバストで高精度な予測値を得る。

2. 伝統的なモデルと機械学習・深層学習的手法によるモデルを組み合わせる。

こちらのワーキングパーティーの今後の展開、展望ですけれども、予測すべき対象と評価指標によって最良のモデルは異なっておりましたので、各モデルの予測値を組み合わせる、つまりアンサンブルですね。それを行うことで、よりロバストで高精度な予測値を得る取り組みが考えられます。緑色で示させていただいておりますものは、「No Free Lunch Theorem」というものでして、簡単に申し上げますと、全ての問題に対して常にベストの予測値を返すようなモデルは存在しないということになります。また、二番目ですけれども、機械学習・深層学習モデルどうしのアンサンブルに限らず、ロバストであった伝統的な手法と、機械学習・深層学習の予測を合わせることで、いいとこ取りの高精度な予測値が出せるのではないかということが考えられます。

ASTIN関連研究会の活動内容のご紹介（2018年度）
データサイエンス関連書籍の翻訳

Predictive Modeling Applications in Actuarial Science
Volume II: Case Studies in Insurance (International Series on Actuarial Science)

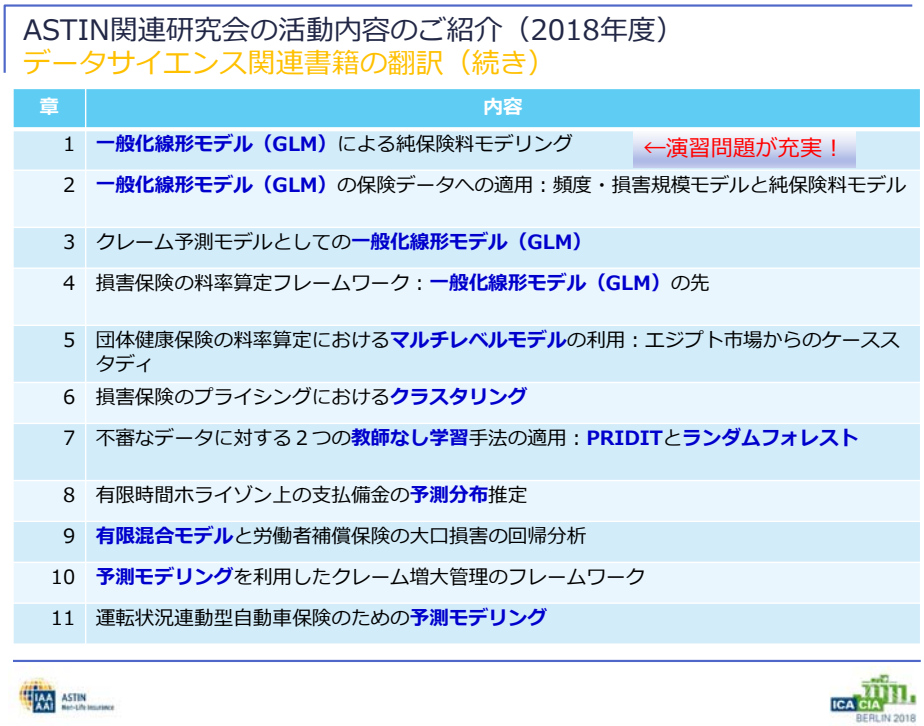


『保険数理における予測モデリングの応用 第II巻』

- ✓ CASの"Predictive Modeling for Actuaries Book Project"で執筆された書籍
- ✓ 原書はアクチュアリー向けに書かれた全２巻からなるシリーズの第２巻で、予測モデリングの基礎を扱ったもの
- ✓ 昨年度（2017年度）、本研究会にて"第１巻"の翻訳作業を行い、eラーニングシステム上で公開
- ✓ 第１巻は線形モデルをベースに予測モデリングの要素（パラメータ推定、変数選択、モデル検証など）について説明
- ✓ 第２巻はより実務向けの内容で、一般化線形モデル（GLM）の応用から、クラスタリングやランダムフォレストなどの手法を用いたケーススタディも記載
  ＊第１章はEDA（探索的データ分析）などの演習問題が充実！
  （HP上でRコードや解説が入手可能）

以上でワーキングパーティーのご説明は終わりにさせていただきますが、最後に、私が委員を務めさせていただいております ASTIN 関連研究会の取り組みに関して、ご報告をさせていただきます。現在、ASTIN 関連研究会では、アクチュアリーの数理領域におけるデータサイエンス関連の書籍の翻訳を行っております。『Predictive Modeling Applications in Actuarial Science Volume II』と題しまして、CAS の Predictive Modeling for Actuaries Book Project で執筆された書籍となりまして、原書はアクチュアリー向けに書かれ、全２巻から成るシリーズの第２巻になります。昨年度、本研究会から第１巻の翻訳作業を行い、e ラーニングシステム上で公開しております。

第１巻は、線形モデルをベースに、予測モデリングの要素を、パラメータ推定等について説明しておりますが、第２巻は、より実務的な内容となっております。一般化線形モデルの応用から、クラスタリングやランダムフォレスト、決定木ベースの手法などを用いたケーススタディも記載しております。



詳しい章ごとの内容は、こちらとなります。演習問題も充実しておりますので、豊富な内容となっております。現在、2019 年の３月完成に向け、翻訳チーム一同で鋭意作業中でございます。翻訳が完成した後には、是非お手に取っていただければと思います。以上、ご清聴ありがとうございました。

【司会】　会場からの質問を受け付けたいところではありますが、終了の時刻となりましたので、以上をもちまして、セッションＢ、「支払備金の見積もりにおける機械学習・深層学習的手法の適用と従来手法との比較（2018 ASTIN WP 参加報告）」を終了します。平松さんに、いま一度大きな拍手をお願いします。