

نام دانشجو :‌

محراب عتیقی

نام استاد :

دکتر امیر تیمور پاینده نجف آبادی

موضوع :‌

تحلیل دیتا صدور و خسارت رشته ثالث یک شرکت بیمه به کمک الگوریتم های یادگیری ماشین و شبکه‌های عصبی

نام درس :

تحليل هاي بيم سنجي مبتني بر ابرداده



زمستان 1403

داشنگاه شهید بهشتی

دانشکده علوم ریاضی

فهرست مطالب

[چکیده 2](#_Toc189155131)

[اهداف 3](#_Toc189155132)

[واژگان کلیدی 4](#_Toc189155133)

[مقدمه 5](#_Toc189155134)

[تعاریف 7](#_Toc189155135)

[مراحل انجام پروژه 11](#_Toc189155136)

[معرفی داده ها 13](#_Toc189155137)

[تحلیل و تفسیر کد ها 15](#_Toc189155138)

[یافته‌ها و نتایج 42](#_Toc189155139)

# چکیده

**در این پروژه، هدف اصلی تحلیل داده‌های صدور و خسارت رشته ثالث یک شرکت بیمه با استفاده از الگوریتم‌های هوش مصنوعی است. داده‌های مورد استفاده شامل اطلاعات مربوط به مشتریان، انواع بیمه‌نامه‌های صادر شده، میزان حق بیمه، تاریخچه خسارت‌ها، میزان پرداختی‌ها و سایر متغیرهای مرتبط بود. با استفاده از روش‌های پیشرفته تحلیل داده و الگوریتم‌های هوش مصنوعی، سعی شد تا متغیر های معنا دار بر خسارت شناسایی شده و از این اطلاعات برای بهبود تصمیم‌گیری‌های شرکت بیمه استفاده شود.**

**الگوریتم‌های هوش مصنوعی نقش حیاتی در تجزیه و تحلیل داده‌ها ایفا می‌کنند، زیرا توانایی پردازش حجم عظیمی از اطلاعات را دارند و الگوهای پیچیده و پنهان را کشف می‌کنند. در پروژه تحلیل داده‌های صدور و خسارت شرکت بیمه، این الگوریتم‌ها با پیش‌بینی دقیق خسارت‌ها و شناسایی مشتریان پرریسک، به شرکت کمک کردند تا ریسک‌های مالی را کاهش داده و تصمیم‌گیری‌های بهینه‌تری داشته باشند. به‌عنوان مثال، مدل‌هایی مانند جنگل تصادفی و شبکه‌های عصبی با تحلیل داده‌های تاریخی، بینش‌هایی ارائه دادند که بهبود فرآیندهای صدور بیمه و مدیریت خسارت را ممکن ساخت.**

**اهمیت الگوریتم‌های هوش مصنوعی در این است که آن‌ها نه‌تنها دقت تحلیل‌ها را افزایش می‌دهند، بلکه سرعت و کارایی فرآیندها را نیز بهبود می‌بخشند. با استفاده از این فناوری‌ها، شرکت‌های بیمه می‌توانند به‌طور خودکار داده‌ها را تحلیل کرده و در زمان واقعی به تغییرات واکنش نشان دهند. این امر منجر به کاهش هزینه‌ها، افزایش سودآوری و ارائه خدمات بهتر به مشتریان می‌شود. در نهایت، هوش مصنوعی به‌عنوان یک ابزار استراتژیک، شرکت‌ها را قادر می‌سازد تا در محیط رقابتی امروز، عملکرد خود را به‌طور چشمگیری بهبود بخشند.**

# اهداف

**هدف اصلی این پروژه، پیش‌بینی احتمال وقوع خسارت برای بیمه‌گذاران بر اساس مشخصات و داده‌های تاریخی آن‌ها است. با استفاده از الگوریتم‌های هوش مصنوعی و یادگیری ماشین، مدلی طراحی می‌شود که بتواند با دقت بالا تشخیص دهد آیا یک بیمه‌گذار با ویژگی‌های خاص در آینده به خسارت دچار می‌شود یا خیر. خروجی این مدل به صورت باینری (1 به معنای وقوع خسارت و 0 به معنای عدم وقوع خسارت) خواهد بود. این پیش‌بینی به شرکت بیمه کمک می‌کند تا ریسک‌های مالی خود را بهتر مدیریت کرده و سیاست‌های بیمه‌ای را به‌طور هدفمند تنظیم کند.**

**در راستای این هدف، تحلیل داده‌های تاریخی شرکت بیمه و شناسایی متغیرهای مؤثر بر وقوع خسارت نیز انجام می‌شود. این تحلیل‌ها به درک بهتر عوامل ریسک و بهبود دقت مدل پیش‌بینی کمک می‌کنند. در نهایت، این پروژه به دنبال ارائه یک ابزار تصمیم‌گیری هوشمند است که بتواند به شرکت بیمه در کاهش هزینه‌های ناشی از خسارت‌های غیرمنتظره و افزایش سودآوری کمک کند.**

**علاوه بر پیش‌بینی وقوع خسارت، این پروژه به دنبال ایجاد یک سیستم طبقه‌بندی هوشمند است که بتواند به‌طور خودکار بیمه‌گذاران را در دو دسته‌ی "با ریسک خسارت" و "بدون ریسک خسارت" دسته‌بندی کند. این سیستم با استفاده از الگوریتم‌های پیشرفته‌ی یادگیری ماشین، مانند درخت تصمیم، جنگل تصادفی و شبکه‌های عصبی، طراحی می‌شود تا بتواند با دقت بالا و بر اساس داده‌های واقعی، پیش‌بینی‌های قابل اعتمادی ارائه دهد. این قابلیت به شرکت بیمه امکان می‌دهد تا منابع خود را به‌طور مؤثرتری تخصیص داده، از مشتریان پرریسک به‌طور ویژه مراقبت کند و در نهایت، از زیان‌های مالی غیرمنتظره جلوگیری نماید. این رویکرد نه‌تنها به بهبود عملکرد مالی شرکت کمک می‌کند، بلکه تجربه‌ی بهتری را نیز برای مشتریان به ارمغان می‌آورد.**

# واژگان کلیدی

* **پیش‌بینی خسارت**
* **الگوریتم‌های هوش مصنوعی**
* **یادگیری ماشین**
* **طبقه‌بندی باینری (1 و ۰)**
* **بیمه‌گذاران پرریسک**
* **تحلیل داده‌های تاریخی**
* **مدیریت ریسک مالی**
* **درخت تصمیم**
* **جنگل تصادفی (Random Forest)**
* **شبکه‌های عصبی**
* **کاهش هزینه‌های خسارت**
* **بهبود تصمیم‌گیری**
* **متغیرهای مؤثر بر خسارت**
* **سیستم پیش‌بینی هوشمند**
* **بهینه‌سازی سیاست‌های بیمه‌ای**

# مقدمه

**در دنیای امروز، صنعت بیمه با حجم عظیمی از داده‌ها مواجه است که شامل اطلاعات مربوط به بیمه‌گذاران، انواع بیمه‌نامه‌های صادر شده، تاریخچه خسارت‌ها و سایر متغیرهای مرتبط می‌شود. این داده‌ها، اگر به درستی تحلیل شوند، می‌توانند بینش‌های ارزشمندی را در اختیار شرکت‌های بیمه قرار دهند تا بتوانند ریسک‌های مالی خود را بهتر مدیریت کرده و تصمیم‌گیری‌های استراتژیک بهتری اتخاذ کنند. با این حال، تحلیل این حجم از داده‌ها با روش‌های سنتی نه‌تنها زمان‌بر است، بلکه ممکن است نتواند الگوهای پیچیده و پنهان موجود در داده‌ها را به‌طور کامل شناسایی کند. اینجاست که فناوری‌های نوین مانند هوش مصنوعی و یادگیری ماشین به کمک صنعت بیمه می‌آیند.**

**هدف این پروژه، استفاده از الگوریتم‌های هوش مصنوعی برای پیش‌بینی احتمال وقوع خسارت در بیمه‌گذاران بر اساس مشخصات و داده‌های تاریخی آن‌ها است. به عبارت دیگر، این پروژه به دنبال پاسخ به این سوال است که آیا یک بیمه‌گذار با ویژگی‌های خاص (مانند سن، نوع بیمه‌نامه، سابقه خسارت و ...) در آینده به خسارت دچار می‌شود یا خیر. خروجی این تحلیل به صورت باینری (1 به معنای وقوع خسارت و 0 به معنای عدم وقوع خسارت) خواهد بود. برای دستیابی به این هدف، از الگوریتم‌های پیشرفته‌ی یادگیری ماشین مانند درخت تصمیم، جنگل تصادفی و شبکه‌های عصبی استفاده می‌شود تا مدلی طراحی شود که بتواند با دقت بالا، بیمه‌گذاران پرریسک را شناسایی کند.**

**این پروژه نه‌تنها به شرکت بیمه کمک می‌کند تا ریسک‌های مالی خود را کاهش دهد، بلکه با ارائه‌ی یک سیستم پیش‌بینی هوشمند، امکان بهینه‌سازی فرآیندهای صدور بیمه‌نامه و مدیریت خسارت را نیز فراهم می‌آورد. در نهایت، این رویکرد منجر به کاهش هزینه‌های ناشی از خسارت‌های غیرمنتظره، افزایش سودآوری و بهبود تجربه‌ی مشتریان خواهد شد. با توجه به اهمیت روزافزون داده‌ها در صنعت بیمه، این پروژه گامی مؤثر در جهت تحول دیجیتال و استفاده از فناوری‌های نوین برای بهبود عملکرد سازمانی محسوب می‌شود.**

# تعاریف

**تعاریف مدل‌های یادگیری ماشین:**

* **GLM (Generalized Linear Model)**

**مدل‌های خطی تعمیم‌یافته (GLM) یک خانواده از مدل‌های آماری هستند که برای تحلیل داده‌هایی استفاده می‌شوند که رابطه‌ی بین متغیرهای مستقل و وابسته ممکن است خطی نباشد. این مدل‌ها می‌توانند برای مسائل رگرسیون و طبقه‌بندی استفاده شوند و از توزیع‌های مختلفی مانند نرمال، دوجمله‌ای و پواسون پشتیبانی می‌کنند.**

* **GLMNET (Generalized Linear Model with Regularization)**

**این مدل نسخه‌ای از مدل‌های خطی تعمیم‌یافته است که از تکنیک‌های regularization مانند لاسو و ریج برای جلوگیری از بیش‌برازش (overfitting) استفاده می‌کند. این مدل به‌ویژه برای داده‌هایی با تعداد متغیرهای زیاد (high-dimensional data) مناسب است.**

* **RPART (Recursive Partitioning and Regression Trees)**

**این مدل یک الگوریتم درخت تصمیم است که برای مسائل رگرسیون و طبقه‌بندی استفاده می‌شود. این مدل با تقسیم‌بندی بازگشتی داده‌ها به زیرمجموعه‌های همگن، یک درخت تصمیم ایجاد می‌کند. این روش ساده و تفسیرپذیر است اما ممکن است به بیش‌برازش منجر شود.**

* **RF (Random Forest)**

**جنگل تصادفی (Random Forest) یک الگوریتم یادگیری مجموعه‌ای (ensemble learning) است که از ترکیب چندین درخت تصمیم ساخته می‌شود. هر درخت بر روی یک نمونه‌ی bootstrap از داده‌ها آموزش می‌بیند و پیش‌بینی نهایی بر اساس میانگین یا رأی اکثریت درخت‌ها انجام می‌شود. این مدل به‌دلیل دقت بالا و مقاومت در برابر بیش‌برازش، بسیار محبوب است.**

* **GBM (Gradient Boosting Machine)**

**ماشین افزایش گرادیان (GBM) یک الگوریتم یادگیری مجموعه‌ای است که از ترکیب چندین مدل ضعیف (معمولاً درخت‌های تصمیم) برای ایجاد یک مدل قوی استفاده می‌کند. این مدل به‌صورت تدریجی و با کاهش خطا (گرادیان) آموزش می‌بیند و برای مسائل رگرسیون و طبقه‌بندی کاربرد دارد.**

* **XGBTREE (Extreme Gradient Boosting Trees)**

**مدلXGBoost یک نسخه‌ی بهینه‌شده و پیشرفته از الگوریتم Gradient Boosting است که از درخت‌های تصمیم استفاده می‌کند. این مدل به‌دلیل سرعت بالا، دقت و قابلیت مدیریت داده‌های بزرگ، بسیار مورد توجه قرار گرفته است. XGBoost از regularization و تکنیک‌های دیگری برای بهبود عملکرد استفاده می‌کند.**

* **SVMRADIAL (Support Vector Machine with Radial Kernel) ماشین بردار پشتیبان (SVM) با کرنل شعاعی (Radial Kernel) یک الگوریتم قدرتمند برای مسائل طبقه‌بندی و رگرسیون است. این مدل با استفاده از توابع کرنل، داده‌ها را به فضای با ابعاد بالاتر منتقل می‌کند تا بتواند مرزهای تصمیم‌گیری غیرخطی ایجاد کند. کرنل شعاعی یکی از پرکاربردترین کرنل‌ها در SVM است.**
* **KNN (K-Nearest Neighbors)**

**الگوریتم K-نزدیک‌ترین همسایه (KNN) یک روش ساده و غیرپارامتری برای طبقه‌بندی و رگرسیون است. این مدل بر اساس فاصله‌ی بین نقاط داده، پیش‌بینی را انجام می‌دهد. برای یک نقطه‌ی جدید، K نزدیک‌ترین همسایه‌ها شناسایی می‌شوند و برچسب کلاس یا مقدار پیش‌بینی شده بر اساس آن‌ها تعیین می‌شود.**

* **PLS (Partial Least Squares)**

**حداقل مربعات جزئی (PLS) یک روش کاهش ابعاد است که برای مسائل رگرسیون و طبقه‌بندی استفاده می‌شود. این مدل با پیدا کردن جهت‌هایی در فضای متغیرهای مستقل که بیشترین همبستگی را با متغیر وابسته دارند، داده‌ها را به ابعاد کم‌تر کاهش می‌دهد. PLS به‌ویژه برای داده‌هایی با تعداد متغیرهای زیاد و هم‌خطی (multicollinearity) مناسب است.**

* **NNET (Neural Network)**

**شبکه‌های عصبی مصنوعی (NNET) مدل‌هایی هستند که از ساختار مغز انسان الهام گرفته‌اند و از لایه‌های متعدد نورون‌ها تشکیل شده‌اند. این مدل‌ها می‌توانند روابط غیرخطی پیچیده را در داده‌ها یاد بگیرند و برای مسائل طبقه‌بندی و رگرسیون استفاده می‌شوند. شبکه‌های عصبی به‌دلیل انعطاف‌پذیری و قدرت بالا، در بسیاری از کاربردهای یادگیری ماشین مورد استفاده قرار می‌گیرند.**

**این مدل‌ها هر کدام مزایا و معایب خاص خود را دارند و انتخاب آن‌ها به نوع مسئله، حجم داده‌ها و اهداف پروژه بستگی دارد.**

# مراحل انجام پروژه

**1- جمع‌آوری و پیش‌پردازش داده‌ها:**

**- داده‌ها شامل اطلاعات ساختاریافته و غیرساختاریافته بودند که نیاز به پاک‌سازی، یکپارچه‌سازی و تبدیل به فرمت مناسب برای تحلیل داشتند.**

**- عملیات پیش‌پردازش شامل حذف داده‌های تکراری، پر کردن مقادیر (Missing Values)، نرمال‌سازی داده‌ها و کدگذاری متغیرهای کیفی انجام شد.**

**2- تحلیل اکتشافی داده‌ها (EDA):**

**- با استفاده از روش‌های آماری و مصورسازی داده‌ها، توزیع متغیرها، همبستگی بین آن‌ها و الگوهای اولیه شناسایی شد.**

**- این مرحله به درک بهتر داده‌ها و شناسایی متغیرهای کلیدی کمک کرد.**

**3- مدل‌سازی با الگوریتم‌های هوش مصنوعی:**

**- از الگوریتم‌های مختلف یادگیری ماشین مانند رگرسیون خطی، درخت تصمیم، جنگل تصادفی (Random Forest)، ماشین بردار پشتیبان (SVM) و شبکه‌های عصبی استفاده شد.**

**- مدل‌ها برای پیش‌بینی میزان خسارت، شناسایی مشتریان پرریسک و پیش‌بینی احتمال وقوع خسارت آموزش داده شدند.**

**- از روش‌های ارزیابی مانند دقت (Accuracy)، دقت طبقه‌بندی (Precision)، بازیابی (Recall) و F1-Score برای ارزیابی عملکرد مدل‌ها استفاده شد.**

**4- بهینه‌سازی و تنظیم مدل‌ها:**

**- با استفاده از تکنیک‌هایی مانند جستجوی شبکه‌ای (Grid Search) و جستجوی تصادفی (Random Search)، پارامترهای مدل‌ها بهینه‌سازی شدند.**

**- از روش‌های کاهش ابعاد مانند PCA(تحلیل مؤلفه‌های اصلی) برای بهبود عملکرد مدل‌ها استفاده شد.**

**5- تحلیل نتایج و ارائه پیشنهادات:**

**- نتایج نشان داد که برخی از متغیرها تأثیر قابل توجهی بر میزان خسارت دارند.**

**- مدل‌های پیش‌بینی کننده با دقت قابل قبولی قادر به پیش‌بینی میزان خسارت و شناسایی مشتریان پرریسک بودند.**

**- بر اساس نتایج، پیشنهاداتی برای بهبود فرآیندهای صدور بیمه‌نامه و مدیریت خسارت ارائه شد.**

# معرفی داده ها

**دیتا استخراج شده از بورد رو صدور و خسارت سیستم بیمه گری شامل 20 اکسل صدور و 3 اکسل خسارت می باشد.**

**تعداد داده های خسارت شامل 2,193,922 مشاهده است.که به عنوان نمونه یکی از دیتاهای صدور شامل متغیر های زیر است:‌**

|  |  |  |
| --- | --- | --- |
| **ردیف** | **نام کاربر ثبت کننده بیمه نامه** | **سابقه سرنشین** |
| **شعبه سرپرست واحد صدور بیمه نامه** | **شماره کامل** | **سابقه مالی / جانی** |
| **استان شعبه سرپرست واحد صدور بیمه نامه** | **کد پرسنلي** | **حق بیمه ثالث اجباری** |
| **سرپرست واحد صدور بیمه نامه** | **نوع بیمه گذار** | **حق بیمه تعدد دیات** |
| **واحد صدور بیمه نامه** | **شرکت بیمه سال قبل** | **حق بیمه مازاد جانی** |
| **استان واحد صدور بیمه نامه** | **شماره بیمه سال قبل** | **حق بیمه مازاد مالی** |
| **تاریخ صدور** | **پلاک** | **حق بیمه حوادث راننده** |
| **نام بیمه گذار** | **شماره موتور** | **حق بیمه صندوق** |
| **کد یونیک بیمه گذار** | **شماره شاسی** | **حق بیمه پایه** |
| **منطقه آموزش و پرورش بیمه گذار** | **سال ساخت خودرو (شمسی)** | **عوارض بهداشت** |
| **بازاریاب** | **سیستم خودرو** | **عوارض رديف 160111 قانون بودجه** |
| **نوع خودرو** | **نوع پلاک** | حق بیمه صندوق (سهم بیمه مرکزی) |
| **گروه خودرو** | **سری چاپ** | مالیات ارزش افزوده |
| **دسته بندی خودرو** | **سریال چاپ** | عوارض ارزش افزوده |
| **مورد استفاده خودرو** | **شماره بايگاني قرارداد** | حق بیمه صندوق (سهم بیمه گر) |
| **تاریخ شروع** | **شماره قرارداد** | خالص حق بیمه |
| **مدت (روز)** | **کد رایانه قرارداد** | خالص حق بيمه + مالیات و عوارض ارزش افزوده |
| **تاریخ پایان** | **پوشش جانی (میلیون ریال)** | خالص حق بيمه - حق بیمه صندوق + مالیات و عوارض ارزش افزوده |
| **کد رایانه** | **پوشش مالی(میلیون ریال)** |  |
| **استان سرپرست واحد صدور بیمه نامه** | **پوشش حوادث راننده(میلیون ریال)** |  |

**دیتا های خسارت شامل 3 فایل اکسل می باشد‌:**

1. **خسارت جانی**
2. **خسارت مالی**
3. **خسارت سرنشین**

**تعداد داده های خسارت شامل 80,157 رکورد است که به عنوان نمونه دیتا خسارت سرنشین شامل متغیر های زیر است:‌**

|  |  |  |
| --- | --- | --- |
| **ردیف** | **شماره بایگانی قرارداد** | **تعهدجاني(ميليون ريال)** |
| **واحد معرف بیمه نامه / الحاقیه** | **شماره قرارداد بیمه نامه / الحاقیه** | **تعهدمالي(ميليون ريال)** |
| **نام بیمه گذار** | **نوع بیمه گذار** | **تعهد حوادث راننده (ميليون ريال)** |
| **کد یونیک بیمه گذار** | **شماره پرونده** | **نوع خسارت** |
| **گروه خودرو** | **سال ساخت خودرو** | **واحد سرپرستی بیمه نامه / الحاقیه** |
| **واحد صدور حواله** | **پلاک** | **آخرین وضعیت پرونده خسارت** |
| **سرپرست واحد صدور حواله** | **تاریخ تشکیل پرونده** | **مبلغ حواله ثالث اجباری پرداختنی** |
| **دسته بندي خودرو** | **کد رایانه خودرو** | **مبلغ حواله ثالث اجباری دریافتنی** |
| **بابت حواله** | **تاريخ حادثه** | **مبلغ حواله ثالث اجباری** |
| **واحد تشکیل پرونده** | **تاریخ اعلام** | مبلغ حواله مازاد پرداختنی |
| **سرپرست واحد تشکیل پرونده** | **کد رایانه پرونده** | مبلغ حواله مازاد دریافتنی |
| **نوع خودرو** | **کاربر ثبت کننده حواله** | مبلغ حواله مازاد |
| **تاريخ صدورحواله** | **نوع گواهینامه مقصر** | مبلغ کل حواله |
| **نوع حواله** | **شماره حواله** | کل مبلغ حواله پرداختنی |
| **واحد صدور بیمه نامه/الحاقیه** | **سابقه مالی** | کل مبلغ حواله دریافتنی |
| **تاریخ شروع** | **سابقه جانی** |  |
| **شماره بیمه نامه** | **سابقه حوادث راننده** |  |
| **تاریخ صدور بیمه نامه** | **مورد استفاده در زمان صدور بیمه نامه** |  |
| **تاریخ پایان** | **مورد استفاده در زمان حادثه** |  |
| **کد رایانه بيمه نامه** | **علت حادثه** |  |

**تحلیل ، بررسی و مدل سازی دیتا ها در نرم افزاز R‌ صورت گرفت که در ادامه به بررسی کد های زده شده در R می پردازیم:‌**

# تحلیل و تفسیر کد ها

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(readxl)  
library(ggplot2)  
library(DBI)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ lubridate 1.9.4 ✔ tibble 3.2.1  
## ✔ purrr 1.0.2 ✔ tidyr 1.3.1  
## ✔ readr 2.1.5

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(data.table)

##   
## Attaching package: 'data.table'  
##   
## The following objects are masked from 'package:lubridate':  
##   
## hour, isoweek, mday, minute, month, quarter, second, wday, week,  
## yday, year  
##   
## The following object is masked from 'package:purrr':  
##   
## transpose  
##   
## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

library(caret) # For machine learning

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(tensorflow) # For TensorFlow backend

##   
## Attaching package: 'tensorflow'  
##   
## The following object is masked from 'package:caret':  
##   
## train

library(keras) # For neural networks  
library(doParallel)

## Loading required package: foreach  
##   
## Attaching package: 'foreach'  
##   
## The following objects are masked from 'package:purrr':  
##   
## accumulate, when  
##   
## Loading required package: iterators  
## Loading required package: parallel

**ابتدا به فراخوانی پکیج های مورد نیاز پرداخته شد.**

########################################################## Policy Data Cleaning  
##read all policies excel files with .xlsx pattern.  
  
file.list = list.files( pattern='\*.xlsx' , recursive = TRUE )  
policies <- lapply(file.list[4:33], read\_excel)  
#  
# # bind each files with row  
all\_policies\_df = bind\_rows(policies)  
  
  
#remove some columns manualy  
  
df\_sodor = as.data.table(  
 all\_policies\_df[,-c(1,2,3,4,10,19,20,23,26,28,29,33,34,35,36,37)]  
 )  
  
df\_sodor =  
 df\_sodor %>%  
 separate(col = `سابقه مالی / جانی`, into = c("c1", "FinanceHistory","LifeHistory"), sep = ":", remove = T)  
df\_sodor = df\_sodor[,-c(which(names(df\_sodor) == "c1"))]  
df\_sodor$FinanceHistory = substr(df\_sodor$FinanceHistory ,  
 start = 1, stop = nchar(df\_sodor$FinanceHistory)-6)  
df\_sodor= as.data.table(df\_sodor)  
# save(df\_sodor , file = "policies\_list.RData")  
# load("policies\_list.RData")  
#find columns with low or very high variance to delete them  
  
# ایجاد یک داده‌فریم نمونه با 40 ستون categorical  
set.seed(123)  
coded\_dfs <- list()  
categorical\_columns <- names(which(  
 sapply(df\_sodor, is.factor) | sapply(df\_sodor, is.character) == T))  
# حلقه برای کددهی به هر ستون و ایجاد داده‌فریم جداگانه  
for (col in categorical\_columns) {  
 # ایجاد ستون کد  
 df\_sodor[, paste0(col, "\_Code") := as.numeric(factor(get(col)))]  
  
 # ایجاد داده‌فریم جداگانه برای این ستون  
 coded\_dfs[[col]] <- df\_sodor[, .(get(col), get(paste0(col, "\_Code")))]  
 coded\_dfs[[col]] <- unique(df\_sodor[, .(get(col), get(paste0(col, "\_Code")))])  
 names(coded\_dfs[[col]]) <- c(col, paste0(col, "\_Code"))  
}  
  
# save(coded\_dfs , file = "coded\_dfs.RData")  
# load("coded\_dfs.RData")  
df\_sodor = as.data.frame(df\_sodor)  
df\_with\_coding = df\_sodor[,which(names(df\_sodor) %notin% categorical\_columns)]  
  
# remove some linearity columns mannulay for example A+B-C = D  
df\_with\_coding = df\_with\_coding[,-c(21,22)]  
names(df\_with\_coding)

## [1] "کد یونیک بیمه گذار" "مدت (روز)"   
## [3] "سال ساخت خودرو (شمسی)" "پوشش جانی (میلیون ریال)"   
## [5] "پوشش مالی(میلیون ریال)" "پوشش حوادث راننده(میلیون ریال)"   
## [7] "حق بیمه ثالث اجباری" "حق بیمه تعدد دیات"   
## [9] "حق بیمه مازاد جانی" "حق بیمه مازاد مالی"   
## [11] "حق بیمه حوادث راننده" "حق بیمه صندوق"   
## [13] "حق بیمه پایه" "عوارض بهداشت"   
## [15] "عوارض رديف 160111 قانون بودجه" "حق بیمه صندوق (سهم بیمه مرکزی)"   
## [17] "مالیات ارزش افزوده" "عوارض ارزش افزوده"   
## [19] "حق بیمه صندوق (سهم بیمه گر)" "خالص حق بیمه"   
## [21] "واحد صدور بیمه نامه\_Code" "استان واحد صدور بیمه نامه\_Code"   
## [23] "تاریخ صدور\_Code" "نام بیمه گذار\_Code"   
## [25] "بازاریاب\_Code" "نوع خودرو\_Code"   
## [27] "گروه خودرو\_Code" "دسته بندی خودرو\_Code"   
## [29] "مورد استفاده خودرو\_Code" "تاریخ شروع\_Code"   
## [31] "تاریخ پایان\_Code" "نام کاربر ثبت کننده بیمه نامه\_Code"  
## [33] "شماره کامل\_Code" "نوع بیمه گذار\_Code"   
## [35] "شرکت بیمه سال قبل\_Code" "پلاک\_Code"   
## [37] "سیستم خودرو\_Code" "نوع پلاک\_Code"   
## [39] "سابقه سرنشین\_Code" "FinanceHistory\_Code"   
## [41] "LifeHistory\_Code"

names(df\_with\_coding) = c("PolicyHolderCode" ,  
 "Duration",  
 "CarProductYear",  
 "SideCover\_MR",  
 "FinanceCover\_MR",  
 "AccidentCover\_MR",  
 "ThirdParty\_Pr",  
 "MultipleBloodMoney\_Pr",  
 "ExcessLife\_Pr",  
 "ExcessFinance\_Pr",  
 "DriverAccident\_Pr",  
 "Pension\_Pr",  
 "Basis\_Pr",  
 "Health\_Complications",  
 "Goverment\_Complications",  
 "CentralInsurance\_Pension\_Pr",  
 "ValueAdded\_Tax",  
 "ValueAdded\_Complications",  
 "Insurer\_Pension\_Pr",  
 "Net\_Pr",  
 "ID\_PolicyLuncher\_Departmant",  
 "ID\_PolicyLuncher\_Province",  
 "ID\_LunchDate",  
 "ID\_PolicyHolderName",  
 "ID\_MarkettingBy",  
 "ID\_AutomobileType",  
 "ID\_AutomobileGroup",  
 "ID\_AutomobileClass",  
 "ID\_AutomobuileUssage",  
 "ID\_StartDate",  
 "ID\_EndDate",  
 "ID\_RegisterUser",  
 "ID\_Policy",  
 "ID\_PolicyHolderType",  
 "ID\_LastInsurer",  
 "ID\_AutomobileZipCode",  
 "ID\_AutomobileSystem",  
 "ID\_AutomobileZipCodeType",  
 "ID\_Passengers\_Claim",  
 "ID\_Finance\_Claim",  
 "ID\_Life\_Claim"  
 )  
# save(df\_with\_coding , file = "df\_with\_coding.RData")  
# load("df\_with\_coding.RData")

**در این کدها، یک فرآیند پاک‌سازی و آماده‌سازی داده‌های مربوط به بیمه‌نامه‌ها (Policy Data) صورت گرفت. مراحل اصلی به شرح زیر است:**

**1- خواندن داده‌ها:**

**- تمام فایل‌های اکسل با پسوند `.xlsx` که حاوی داده‌های بیمه‌نامه‌ها بودند، خوانده شدند و در یک لیست ذخیره شدند.**

**- سپس این فایل‌ها به یک داده‌فریم واحد (`all\_policies\_df`) تبدیل شدند.**

**2- حذف ستون‌های غیرضروری:**

**- برخی ستون‌ها که مورد نیاز نبودند، به صورت دستی حذف شدند.**

**3- تجزیه و اصلاح داده‌ها:**

**- ستون `سابقه مالی / جانی` به دو ستون جداگانه (`FinanceHistory` و `LifeHistory`) تقسیم شد و بخش‌های اضافی از این ستون‌ها حذف شدند.**

**4- کدگذاری داده‌های کیفی:**

**- ستون‌های کیفی (کاراکتر یا فاکتور) شناسایی و به صورت عددی کدگذاری شدند.**

**- برای هر ستون کیفی، یک ستون جدید با پسوند `\_Code` ایجاد شد که حاوی کدهای عددی مربوطه بود.**

**5- حذف ستون‌های با واریانس کم یا زیاد:**

**- ستون‌هایی که دارای واریانس کم یا زیاد بودند و یا باعث ایجاد خطی‌بودن (linearity) می‌شدند، حذف شدند.**

**6- تغییر نام ستون‌ها:**

**- نام ستون‌ها به انگلیسی تغییر داده شد تا خوانایی و یکپارچگی داده‌ها بهبود یابد.**

**7- ذخیره‌سازی داده‌ها:**

**- داده‌های نهایی در قالب فایل‌های `.RData` ذخیره شدند تا در مراحل بعدی تحلیل مورد استفاده قرار گیرند.**

**در نهایت، یک داده‌فریم تمیز و آماده برای تحلیل‌های بعدی ایجاد کردید که شامل داده‌های عددی و کدگذاری‌شده است. این داده‌ها می‌توانند برای آموزش مدل‌های یادگیری ماشین و پیش‌بینی خسارت استفاده شوند.**

####################################################Claim Data Cleaning  
file.list = list.files( pattern='\*.xlsx' , recursive = TRUE )  
 Claims <- lapply(file.list[1:3], read\_excel)  
 names(Claims[[1]])[54] = "LifeLoss\_Value"  
 names(Claims[[2]])[54] = "PassengerLoss\_Value"  
 names(Claims[[3]])[54] = "FinanceLoss\_Value"  
   
# bind each files with row  
 all\_claims\_df = bind\_rows(Claims)  
  
#remove some columns manualy  
  
 df\_claims = as.data.table(  
 all\_claims\_df[,c(17,54,56,57)]  
 )  
 df\_claims = as.data.frame(df\_claims)  
names(df\_claims)[1] = c("ID\_Policy")  
for(i in 2:4){  
 df\_claims[,i] = ifelse(is.na(df\_claims[,i]) , 0 , df\_claims[,i])  
}

######################################################## Merging Data  
library(dplyr)  
library(readxl)  
library(ggplot2)  
library(DBI)  
library(tidyverse)  
library(data.table)  
  
# load("df\_with\_coding.RData")  
# load("Claims\_list.RData")  
# load("coded\_dfs.RData")  
names(coded\_dfs[["شماره کامل"]])[1] = "ID\_Policy"  
# coded\_dfs[["شماره کامل"]]$ID\_Policy = as.character(coded\_dfs[["شماره کامل"]]$ID\_Policy)  
claims\_with\_code = left\_join(df\_claims , coded\_dfs[["شماره کامل"]] , by = "ID\_Policy",keep = F )  
claims\_with\_code = claims\_with\_code[,-c(1)] #remove ID\_Ploicy column  
names(claims\_with\_code)[4] = "ID\_Policy"  
FinalDf = left\_join(df\_with\_coding , claims\_with\_code , by = "ID\_Policy")  
# save(FinalDf , file = "FinalDf.RData")

**در این کد مراحل زیر صورت گرفته است:**

**1- خواندن فایل‌ها:**

**- فایل‌های اکسل با پسوند `.xlsx` از دایرکتوری جاری و زیردایرکتوری‌ها خوانده شده‌اند.**

**- سه فایل اول به لیست `Claims` اضافه شده‌اند.**

**2- تغییر نام ستون‌ها:**

**- ستون 54 در هر یک از سه فایل به ترتیب به `LifeLoss\_Value`، `PassengerLoss\_Value` و `FinanceLoss\_Value` تغییر نام داده شده است.**

**3- ادغام داده‌ها:**

**- داده‌های سه فایل به صورت سطری (row-wise) با هم ادغام شده‌اند و در `all\_claims\_df` ذخیره شده‌اند.**

**4- حذف ستون‌ها:**

**- تنها ستون‌های 17، 54، 56 و 57 از داده‌های ادغام‌شده انتخاب شده‌اند و در `df\_claims` ذخیره شده‌اند.**

**- نام ستون اول به `ID\_Policy` تغییر یافته است.**

**- مقادیر `NA` در ستون‌های 2 تا 4 با صفر جایگزین شده‌اند.**

**5- ادغام با داده‌های کدگذاری‌شده:**

**- داده‌های `df\_claims` با داده‌های کدگذاری‌شده (`coded\_dfs)"شماره کامل بر اساس ستون `ID\_Policy` ادغام شده‌اند.**

**- ستون `ID\_Policy` از داده‌های ادغام‌شده حذف شده و نام ستون چهارم به `ID\_Policy` تغییر یافته است.**

**6- ادغام نهایی:**

**- داده‌های `df\_with\_coding` با داده‌های ادغام‌شده (`claims\_with\_code`) بر اساس `ID\_Policy` ادغام شده‌اند و نتیجه در `FinalDf` ذخیره شده است.**

########################################################### CV Modeling  
# cl <- makePSOCKcluster(4) # تعداد هسته‌های پردازنده (مثلاً 4)  
# registerDoParallel(cl)  
#load("FinalDf.RData")  
for(i in 42:44){  
 FinalDf[,i] = ifelse(is.na(FinalDf[,i]) , 0 , FinalDf[,i])  
}  
FinalDf$HaveLoss =   
 ifelse(FinalDf$LifeLoss\_Value +  
 FinalDf$PassengerLoss\_Value +  
 FinalDf$FinanceLoss\_Value > 0 ,1,0)  
FinalDf$HaveLoss = as.factor(FinalDf$HaveLoss)  
#remove some columns for na values and other loss types.  
  
# Check for missing values in each column  
missing\_values <- colSums(is.na(FinalDf))  
  
FinalDf = FinalDf[,-c(42:44)]  
FinalDf = FinalDf[,-c(which(colnames(FinalDf) %in% names(which(missing\_values>0 ))))]  
split\_data <- function(data, train\_percentage) {  
 # Ensure the train\_percentage is between 0 and 1  
 if (train\_percentage < 0 || train\_percentage > 1) {  
 stop("train\_percentage must be between 0 and 1")  
 }  
 # Calculate the number of rows for the training set  
 n <- nrow(data)  
 n\_train <- floor(train\_percentage \* n)  
   
 # Randomly sample the indices for the training set  
 train\_indices <- sample(1:n, n\_train)  
   
 # Create the training and test sets  
 train\_set <- data[train\_indices, ]  
 test\_set <- data[-train\_indices, ]  
   
 # Return the training and test sets as a list  
 return(list(train = train\_set, test = test\_set))  
}  
set.seed(123)  
# Assuming finalDf is your dataset and you want 80% for training  
result <- split\_data(FinalDf, train\_percentage = 0.8)  
train\_set <- result$train  
test\_set <- result$test  
  
######  
dim(train\_set)

## [1] 1755137 36

head(train\_set)

## PolicyHolderCode Duration CarProductYear SideCover\_MR FinanceCover\_MR  
## 415 210800.4 365 1377 350 20  
## 463 214757.9 365 1382 350 10  
## 179 219846.8 365 1383 400 20  
## 526 221096.4 365 1364 400 20  
## 195 211513.5 4 1380 350 10  
## 938 267169.1 365 1384 350 10  
## ThirdParty\_Pr MultipleBloodMoney\_Pr ExcessLife\_Pr ExcessFinance\_Pr  
## 415 208250 0 818332 272777  
## 463 189000 0 896898 298966  
## 179 169575 0 981349 327116  
## 526 378250 0 2165756 721919  
## 195 12250 0 58454 19485  
## 938 232750 0 1110609 370203  
## DriverAccident\_Pr Pension\_Pr Basis\_Pr Health\_Complications  
## 415 119000 0 1702033 141837  
## 463 72000 0 1748238 145687  
## 179 104500 0 1899050 158255  
## 526 93500 0 4031311 335943  
## 195 4000 0 113029 9420  
## 938 76000 0 2147476 178957  
## Goverment\_Complications CentralInsurance\_Pension\_Pr ValueAdded\_Tax  
## 415 0 0 0  
## 463 0 0 0  
## 179 0 0 0  
## 526 0 0 0  
## 195 0 0 0  
## 938 0 0 0  
## ValueAdded\_Complications Insurer\_Pension\_Pr Net\_Pr  
## 415 0 0 1560196  
## 463 0 0 1602551  
## 179 0 0 1740795  
## 526 0 0 3695368  
## 195 0 0 103609  
## 938 0 0 1968519  
## ID\_PolicyLuncher\_Departmant ID\_PolicyLuncher\_Province ID\_LunchDate  
## 415 1114 8 167  
## 463 1114 8 167  
## 179 776 17 149  
## 526 1114 8 260  
## 195 1163 8 153  
## 938 1163 8 403  
## ID\_AutomobileType ID\_AutomobileGroup ID\_AutomobuileUssage ID\_StartDate  
## 415 4078 3 67 189  
## 463 969 3 67 189  
## 179 5088 2 47 168  
## 526 2981 2 19 303  
## 195 1038 3 57 172  
## 938 1069 3 67 472  
## ID\_EndDate ID\_RegisterUser ID\_Policy ID\_PolicyHolderType  
## 415 194 444 1514 1  
## 463 194 444 1557 1  
## 179 173 680 1977 2  
## 526 307 250 5915 1  
## 195 1 2173 787 1  
## 938 476 2262 3582 2  
## ID\_AutomobileZipCode ID\_AutomobileZipCodeType ID\_Passengers\_Claim  
## 415 3539 10 2  
## 463 3539 10 9  
## 179 2364 11 23  
## 526 537 12 13  
## 195 660 10 4  
## 938 657 10 23  
## ID\_Finance\_Claim ID\_Life\_Claim HaveLoss  
## 415 2 2 0  
## 463 9 9 0  
## 179 23 23 0  
## 526 13 13 0  
## 195 4 4 0  
## 938 23 23 0

**این کد یک فرآیند پیش‌پردازش داده‌ها (Data Preprocessing) و تقسیم داده‌ها به دو بخش آموزش (Train) و آزمون (Test) را انجام می‌دهد. در ادامه به طور مختصر مراحل و خروجی‌ها توضیح داده می‌شوند:**

**مراحل انجام شده:**

**1- مدیریت مقادیر گم‌شده (NA):**

**- مقادیر گم‌شده در ستون‌های 42 تا 44 با `0` جایگزین شده‌اند.**

**2- تعریف متغیر هدف (HaveLoss)**

**- یک متغیر جدید به نام `HaveLoss` ایجاد شده است که نشان می‌دهد آیا مجموع مقادیر `LifeLoss\_Value`، `PassengerLoss\_Value` و `FinanceLoss\_Value` بزرگ‌تر از صفر است یا خیر.**

**- اگر مجموع این مقادیر بز رگ‌تر از صفر باشد، HaveLoss` برابر `1` (دارای ضرر) و در غیر این صورت برابر `0` (بدون ضرر) خواهد بود.**

**- این متغیر به عنوان یک فاکتور (عامل) در نظر گرفته شده است.**

**3- حذف ستون‌های با مقادیر گم‌شده:**

**- ستون‌هایی که دارای مقادیر گم‌شده (NA) هستند، شناسایی و حذف شده‌اند.**

**4- تقسیم داده‌ها به دو بخش آموزش و آزمون:**

**- داده‌ها به دو بخش 80 درصد داده آموزشی و 20 درصد داده تست افراز شدند.**

**داده‌های آموزش شامل 1755137 سطر و 36 ستون است. این بدان معناست که 80% از داده‌ها (یعنی 1755137 سطر از 2193922 سطر) برای آموزش مدل استفاده می‌شوند.**

**در ادامه 6 سطر اول از داده‌های آموزش نمایش داده شده است. هر سطر شامل مقادیر مربوط به 36 ستون است.**

**- ستون HaveLoss نشان می‌دهد که آیا بیمه‌گذار در این سطر دارای ضرر (`1`) بوده یا خیر (`0`).**

# Separate features (X) and target (y) for training and testing sets  
X\_train <- train\_set[, -c(18:dim(train\_set)[2])] # All columns except the last one  
y\_train <- train\_set[, c(dim(train\_set)[2])] # Last column (TotalLoss)  
  
X\_test <- test\_set[, -c(18:dim(test\_set)[2])] # All columns except the last one  
y\_test <- test\_set[, c(dim(test\_set)[2])] # Last column (TotalLoss)  
# Scale/normalize the features  
preprocess\_params <- preProcess(X\_train, method = c("center", "scale"))  
X\_train\_scaled <- predict(preprocess\_params, X\_train)  
X\_test\_scaled <- predict(preprocess\_params, X\_test)  
  
  
X\_train\_scaled = cbind(X\_train\_scaled, train\_set[,18:32])  
X\_test\_scaled = cbind(X\_test\_scaled, test\_set[,18:32])  
  
train\_control <- trainControl(  
 method = "cv", # Cross-validation  
 number = 5, # 5-fold CV  
 savePredictions = "final",  
 allowParallel = TRUE,  
 #verboseIter = TRUE # Show progress updates  
   
)  
  
# Define a list of models to train  
models <- c(  
 #"lm", # Linear Regression  
 "glm", # Logistic Regression  
 "glmnet", # Ridge/Lasso Regression  
 "rpart", # Decision Trees  
 "rf", # Random Forests  
 "gbm", # Gradient Boosting Machines  
 "xgbTree", # XGBoost (Gradient Boosting)  
 "svmRadial", # Support Vector Machines (Radial Kernel)  
 "knn", # k-Nearest Neighbors  
 "pls", # Principal Component Regression  
 #"lda", # Linear Discriminant Analysis  
 #"qda", # Quadratic Discriminant Analysis  
 #"naive\_bayes", # Naive Bayes  
 "nnet" # Neural Networks  
)

**این کد در R برای پیش‌پردازش داده‌ها، آموزش مدل‌های مختلف یادگیری ماشین و ارزیابی آن‌ها با استفاده از اعتبارسنجی متقابل (Cross-Validation) نوشته شده است. در ادامه به طور خلاصه مراحل انجام شده و تحلیل خروجی‌ها آورده شده است.**

**مراحل انجام شده:**

**1- جداسازی متغیر های کمکی و پاسخ**

**2- نرمال‌سازی داده‌ها:**

**- داده‌های آموزشی و آزمون با استفاده از روش‌های `center` و `scale` نرمال‌سازی شده‌اند.**

**3- اضافه کردن ستون‌های ۱۸ تا ۳۲ به داده‌های نرمال‌شده:**

**- این ستون‌ها پس از نرمال‌سازی به داده‌های آموزشی و آزمون اضافه شده‌اند.**

**تنظیم پارامترهای اعتبارسنجی متقابل:**

**- از روش ۵-فولد Cross-Validation برای ارزیابی مدل‌ها استفاده شده است.**

**4- تعریف مدل‌های یادگیری ماشین:**

**مدل های آموزشی مورد استفاده شده:**

* **Glm**
* **Glmnet**
* **Rpart**
* **Rf**
* **Gbm**
* **Xgbtree**
* **Svmradial**
* **Knn**
* **Pls**
* **nnet**

**تحلیل خروجی:**

**- اعتبارسنجی متقابل (CV): با استفاده از ۵-فولد CV، مدل‌ها بر روی زیرمجموعه‌های مختلف داده‌های آموزشی ارزیابی شده‌اند. این روش به کاهش واریانس و افزایش قابلیت تعمیم مدل کمک می‌کند.**

**-مدل‌های آموزش‌دیده: هر مدل با توجه به معیارهای ارزیابی (مانند دقت، خطا و ...) بر روی داده‌های آزمون ارزیابی می‌شود. مدل‌هایی مانند Random Forests (rf) و XGBoost (xgbTree) معمولاً عملکرد بهتری در پیش‌بینی دارند.**

**-نتایج: پس از آموزش مدل‌ها، می‌توان با استفاده از معیارهایی مانند `RMSE` برای رگرسیون یا `Accuracy`برای طبقه‌بندی بهترین مدل را انتخاب کرد.**

# Train and evaluate all models  
results <- list()  
test\_errors <- data.frame(Model = character(), RMSE = numeric(), R2 = numeric(), MAE = numeric(), stringsAsFactors = FALSE)  
  
for (model in models) {  
 set.seed(123) # For reproducibility  
 print(paste("Training model:", model))  
   
 # Train the model  
 fit <- caret::train(  
 x = X\_train\_scaled, # Features  
 y = y\_train, # Target variable  
 method = model, # Model type  
 trControl = train\_control  
 )  
   
 # Store the results  
 results[[model]] <- fit  
   
 # Predict on the test set  
 predictions <- predict(fit, newdata = X\_test\_scaled)  
   
 # Calculate test error metrics  
 if (is.factor(y\_test)) { # Classification  
 cm <- confusionMatrix(predictions, y\_test)  
 accuracy <- cm$overall["Accuracy"]  
 kappa <- cm$overall["Kappa"]  
 test\_errors <- rbind(test\_errors, data.frame(Model = model, Accuracy = accuracy, Kappa = kappa))  
 } else { # Regression  
 rmse <- sqrt(mean((as.numeric(predictions) - as.numeric(y\_test))^2))  
 r2 <- cor(as.numeric(predictions), as.numeric(y\_test))^2  
 mae <- mean(abs(as.numeric(predictions) - as.numeric(y\_test)))  
 test\_errors <- rbind(test\_errors, data.frame(Model = model, RMSE = rmse, R2 = r2, MAE = mae))  
 }  
}

# Print test errors  
print(test\_errors)

## Model Accuracy Kappa  
## Accuracy glm 0.840 -0.019108280  
## Accuracy1 glmnet 0.845 -0.009771987  
## Accuracy2 rpart 0.850 0.000000000  
## Accuracy3 rf 0.865 0.158878505  
## Accuracy4 gbm 0.835 0.014925373  
## Accuracy5 xgbTree 0.850 0.122807018  
## Accuracy6 svmRadial 0.850 0.000000000  
## Accuracy7 knn 0.845 -0.009771987  
## Accuracy8 pls 0.850 0.000000000  
## Accuracy9 nnet 0.850 0.000000000

**هدف اصلی این کد، آموزش مدل‌های مختلف بر روی داده‌های آموزشی و سپس ارزیابی عملکرد آن‌ها بر روی داده‌های تست است. در ادامه، مراحل اصلی کد و خروجی‌ها به طور خلاصه توضیح داده می‌شوند:**

**مراحل اصلی کد:**

**1. تعریف لیست‌ها و داده‌ها:**

**- یک لیست به نام results برای ذخیره نتایج مدل‌ها و یک دیتافریم به نام test\_errors برای ذخیره خطاهای تست (مانند RMSE، R2، MAE برای رگرسیون و Accuracy و Kappa برای طبقه‌بندی) تعریف شده است.**

**2. آموزش مدل‌ها:**

**- یک حلقه for برای آموزش مدل‌های مختلف اجرا می‌شود. مدل‌هایی که آموزش داده می‌شوند شامل glm`، `glmnet`، `rpart`، `rf`، `gbm و xgbTree هستند.**

**- برای هر مدل، ابتدا مدل بر روی داده‌های آموزشی آموزش داده می شود.**

**- سپس مدل آموزش‌دیده بر روی داده‌های تستX\_test\_scaled اعمال می‌شود و پیش‌بینی‌ها انجام می‌گیرد.**

**3. محاسبه خطاهای تست:**

**- اگر مسئله طبقه‌بندی باشد (یعنی y\_test فاکتور باشد)، دقت (Accuracy) و کاپا (Kappa) محاسبه می‌شود.**

**- اگر مسئله رگرسیون باشد، خطاهای RMSE، R2 و MAE محاسبه می‌شوند.**

**4. ذخیره نتایج:**

**- نتایج هر مدل در لیست results و خطاهای تست در دیتافریم test\_errors ذخیره می‌شوند.**

**5. چاپ خطاهای تست:**

**- در نهایت، خطاهای تست برای هر مدل چاپ می‌شود.**

**تفسیر خروجی:**

**- دقت مدل‌ها (Accuracy)‌: دقت مدل‌ها در طبقه‌بندی داده‌های تست بین 0.835 تا 0.865 متغیر است. مدل rf (Random Forest) با دقت 0.865 بهترین عملکرد را دارد.**

**- کاپا (Kappa) : مقدار کاپا برای مدل‌ها بین -0.019 تا 0.158 متغیر است. مدل rf با کاپای 0.158 بهترین عملکرد را در این زمینه دارد.**

**- خطاهای رگرسیون: اگر مسئله رگرسیون بود، خطاهای RMSE، R2 و MAE برای هر مدل محاسبه می‌شد، اما در اینجا فقط نتایج طبقه‌بندی نشان داده شده است.**

**نتیجه‌گیری:**

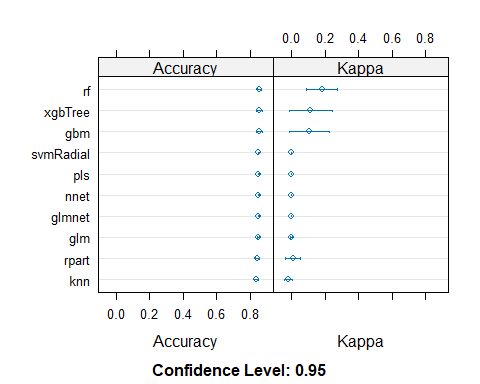
**- مدل rf (Random Forest) بهترین عملکرد را در بین مدل‌های آموزش‌دیده دارد، زیرا هم دقت و هم کاپای آن بالاتر از سایر مدل‌ها است.**

**- مدل‌های دیگر مانند xgbTree و rpart نیز عملکرد نسبتاً خوبی دارند، اما به پای مدل rf نمی‌رسند.**

# Compare model performance (cross-validation results)  
comparison <- resamples(results)  
summary(comparison)

##   
## Call:  
## summary.resamples(object = comparison)  
##   
## Models: glm, glmnet, rpart, rf, gbm, xgbTree, svmRadial, knn, pls, nnet   
## Number of resamples: 5   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## glm 0.84375 0.85000 0.8500 0.85000 0.85000 0.85625 0  
## glmnet 0.85000 0.85000 0.8500 0.85125 0.85000 0.85625 0  
## rpart 0.82500 0.84375 0.8500 0.84375 0.85000 0.85000 0  
## rf 0.84375 0.85625 0.8625 0.85875 0.86250 0.86875 0  
## gbm 0.83125 0.85000 0.8625 0.85625 0.86875 0.86875 0  
## xgbTree 0.83125 0.85625 0.8625 0.85625 0.86250 0.86875 0  
## svmRadial 0.85000 0.85000 0.8500 0.85125 0.85000 0.85625 0  
## knn 0.82500 0.83750 0.8375 0.84000 0.85000 0.85000 0  
## pls 0.85000 0.85000 0.8500 0.85125 0.85000 0.85625 0  
## nnet 0.85000 0.85000 0.8500 0.85125 0.85000 0.85625 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu.  
## glm -0.01214575 0.00000000 0.00000000 -0.002429150 0.00000000  
## glmnet 0.00000000 0.00000000 0.00000000 0.000000000 0.00000000  
## rpart -0.04477612 0.00000000 0.00000000 0.008524119 0.04214559  
## rf 0.09090909 0.14022518 0.17910448 0.180942715 0.20415225  
## gbm -0.03448276 0.05511811 0.14022518 0.107073461 0.17910448  
## xgbTree -0.03448276 0.07221929 0.11877395 0.114395719 0.17910448  
## svmRadial 0.00000000 0.00000000 0.00000000 0.000000000 0.00000000  
## knn -0.04477612 -0.03431129 -0.02362205 -0.020541891 0.00000000  
## pls 0.00000000 0.00000000 0.00000000 0.000000000 0.00000000  
## nnet 0.00000000 0.00000000 0.00000000 0.000000000 0.00000000  
## Max. NA's  
## glm 0.00000000 0  
## glmnet 0.00000000 0  
## rpart 0.04525112 0  
## rf 0.29032258 0  
## gbm 0.19540230 0  
## xgbTree 0.23636364 0  
## svmRadial 0.00000000 0  
## knn 0.00000000 0  
## pls 0.00000000 0  
## nnet 0.00000000 0

# Visualize model performance  
dotplot(comparison)



1. **مقایسه دقت (Accuracy) مدل‌ها:**

* **مدل های rf،‌ gbm و xgbtree بالاترین میانگین دقت را دارند.**
* **مدل های knn و rpart کمترین دقت را دارند.**
* **سایر مدل ها glm، glmnet، nnet، pls، svmradial تقریبا دقت مشابهی دارند.**

1. **مقایسه کاپا (شاخص توافق کلاس بندی) مدل ها:**

* **Rf بهترین مقدار Kappa را دارد.که نشان‌دهنده توانایی بالاتر آن در پیش‌بینی کلاس‌های درست است.**
* **Gbm و xgbtree نیز عملکرد خوبی دارند.**
* **بقیه مدل ها به خصوص nnet, pls, svmradial, glmnet , glm مقدار کاپا نزدیک به صفر یا حتی منفی دارند، که نشان می‌دهد عملکرد آن‌ها تفاوت زیادی با یک مدل تصادفی ندارد.**

**نتیجه‌گیری کلی:**

**• اگر هدف، بیشترین دقت باشد، مدل‌های rf, gbm, و xgbTree انتخاب‌های بهتری هستند.**

**• اگر به‌دنبال مدلی هستید که علاوه بر دقت بالا، قابلیت تمایز بهتری بین کلاس‌ها داشته باشد، rf بهترین گزینه است.**

**• مدل‌های glm, svmRadial, pls, nnet احتمالاً برای این مسئله مناسب نیستند، زیرا مقدار Kappa آن‌ها صفر است، یعنی پیش‌بینی‌های آن‌ها تفاوت خاصی با یک مدل تصادفی ندارد.**

# یافته‌ها و نتایج

**در این پروژه، با استفاده از الگوریتم‌های مختلف یادگیری ماشین، مدل‌هایی برای پیش‌بینی وقوع خسارت بیمه‌ای طراحی و ارزیابی شدند. نتایج نشان داد که مدل‌های Random Forest (rf) , Gradient Boosting Machine (gbm), و XGBoost (xgbTree) از نظر دقت پیش‌بینی، عملکرد بهتری نسبت به سایر مدل‌ها دارند. این مدل‌ها توانستند با دقت بالایی وقوع خسارت را پیش‌بینی کنند، که نشان‌دهنده‌ی قابلیت آن‌ها در یادگیری الگوهای پیچیده از داده‌های تاریخی است. به‌ویژه، مدل Random Forest به دلیل قابلیت تمایز بهتر بین کلاس‌ها، به عنوان بهترین گزینه برای این مسئله شناسایی شد. این مدل نه‌تنها دقت بالایی دارد، بلکه توانایی تشخیص دقیق‌تری بین بیمه‌گذاران پرریسک و کم‌ریسک را نیز فراهم می‌کند.**

**از سوی دیگر، مدل‌هایی مانند Generalized Linear Model (glm) ، Support Vector Machine with Radial Kernel (svmRadial) ، Partial Least Squares (pls) و Neural Network (nnet) عملکرد ضعیف‌تری داشتند. مقدار Kappa این مدل‌ها صفر بود، که نشان می‌دهد پیش‌بینی‌های آن‌ها تفاوت معناداری با یک مدل تصادفی ندارد. این نتیجه بیانگر آن است که این مدل‌ها برای مسئله‌ی پیش‌بینی وقوع خسارت بیمه‌ای مناسب نیستند و نمی‌توانند الگوهای مؤثر را از داده‌ها استخراج کنند. بنابراین، تمرکز اصلی در این پروژه بر روی مدل‌های rf، gbm و xgbTree قرار گرفت.**

**در نهایت، این پروژه موفق به طراحی یک سیستم طبقه‌بندی هوشمند شد که می‌تواند بیمه‌گذاران را به‌طور خودکار در دو دسته‌ی "با ریسک خسارت" و "بدون ریسک خسارت" دسته‌بندی کند. این سیستم با استفاده از مدل‌های پیشرفته‌ی یادگیری ماشین، مانند Random Forest و XGBoost، قادر است پیش‌بینی‌های دقیق و قابل اعتمادی ارائه دهد. این قابلیت به شرکت بیمه کمک می‌کند تا منابع خود را به‌طور مؤثرتری مدیریت کرده، از مشتریان پرریسک به‌طور ویژه مراقبت کند و از زیان‌های مالی غیرمنتظره جلوگیری نماید. این رویکرد نه‌تنها به بهبود عملکرد مالی شرکت کمک می‌کند، بلکه تجربه‌ی بهتری را نیز برای مشتریان به ارمغان می‌آورد.**

**پایان.**