گزارش کار تمرین 13

توسط: کتایون کبرائی

استاد مربوطه: استاد فراهانی

چکیده:

در این قسمت از سری دوم تمرینات تلاش شده است با به کارگیری متدهای مختلف یادگیری ماشین بهترین مدل های یادگیری برای Vehicle Insurance Claim Fraud Detection dataset به دست اورده شود. ابتدا برای در این فرایند پیش پردازش هایی روی مدل انجام میشود تا دیتا به کاربردی ترین شکل خود دراید. در این راستا دیتا با نمودار های مرتبط visualize میشود تا بهترین نتیجه گیری از ان بدست اید. سپس معروف ترین مدل های classification روی این dataset تست میشود و تلاش میشود با استفاده از روش هایی مثل نمودار roc و غیره بهترین‌شان بدست اید. حتی برای نتیجه‌گیری بهتر برای بهترین مدل از روش‌های cross_validation نیز استفاده خواهد شد. در ادامه فرایند به بررسی دیتاهای imbalance پرداخته میشود. برای کنترل این دیتا نیز روش هایی مثل OverSampling و UnderSampling به کار گرفته میشود. در نهایت هم برای اینکه برترین مدل های انتخاب بهترین نتیجه را به دهند تکنیک های boosting برای مدل به کار گرفته میشود تا بالاترین دقت به دست اید. در ادامه به بررسی جزئیات تمامی این فرایندها میپردازیم.

مقدمه:

دیتاستی که در این تمرین با ان کار خواهد شد Vehicle Insurance Claim Fraud Detection است. در توضیح این دیتاست باید گفت لیستی از تمام تصادفاتی رخ داده در تاریخی مشخص جمع‌اوری شده است و در نهایت تلاش بر این است که مدل با این دیتاست اموزش داده شود که تشخیص داده شود فرد درخواست داده شده برای بیمه ایا مرتکب ادعای تقلبی شده است یا خیر. از انجایی که این یک مسئله binary classification است نیاز است تا معروف ترین مدل های این دسته مسائل یعنی linear regression یا random forest و ... روی ان تست شود. پس برای تشخیص بهترین مدل یادگیری پس از اماده‌سازی داده ها برای پردازش شدن، به مقایسه این مدل ها پرداخته میشود و پردازش شان بهبود داده میشود تا در نهایت بهترین مدل ممکن پیدا شود.

بخش اول مسئله: بررسی کل دیتاست و اماده سازی داده (Performing exploratory data analysis)

ابتدا ساختار کلی دیتاست یعنی تمامی فیچرهای مسئله، تعداد کل نمونه ها، ساختار هر فیچر و چند نمونه تصادفی بررسی میشود.

برای این کار نیاز است با استفاده از تابع `read_csv()` فایل مورد نظر بارگذاری میشود و در یک dataframe ذخیره میشود.

سپس با استفاده از ویژگی `columns` تمامی فیچرهای دیتاست چاپ میشود که شامل `Month, WeekOfMonth, DayOfWeek, Make, AccidentArea, DayOfWeekClaimed, MonthClaimed, WeekOfMonthClaimed, Sex, MaritalStatus, Age, Fault, PolicyType, VehicleCategory, VehiclePrice, FraudFound_P, PolicyNumber, RepNumber, Deductible, DriverRating, Days_Policy_Accident, Days_Policy_Claim, PastNumberOfClaims, AgeOfVehicle, AgeOfPolicyHolder, PoliceReportFiled, WitnessPresent, AgentType, NumberOfSuppliments, AddressChange_Claim, NumberOfCars, Year, BasePolicy` خواهد بود. سپس با استفاده از تابع `info()` جنس هر یک از این ویژگی ها بدست خواهد امد.

```
#   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Month                 15420 non-null  object
 1   WeekOfMonth           15420 non-null  int64
 2   DayOfWeek             15420 non-null  object
 3   Make                  15420 non-null  object
 4   AccidentArea          15420 non-null  object
 5   DayOfWeekClaimed      15420 non-null  object
 6   MonthClaimed          15420 non-null  object
 7   WeekOfMonthClaimed    15420 non-null  int64
 8   Sex                   15420 non-null  object
 9   MaritalStatus         15420 non-null  object
 10  Age                   15420 non-null  int64
 11  Fault                 15420 non-null  object
 12  PolicyType            15420 non-null  object
 13  VehicleCategory       15420 non-null  object
 14  VehiclePrice          15420 non-null  object
 15  FraudFound_P          15420 non-null  int64
 16  PolicyNumber          15420 non-null  int64
 17  RepNumber             15420 non-null  int64
 18  Deductible            15420 non-null  int64
 19  DriverRating          15420 non-null  int64
 20  Days_Policy_Accident  15420 non-null  object
 21  Days_Policy_Claim     15420 non-null  object
 22  PastNumberOfClaims    15420 non-null  object
 23  AgeOfVehicle          15420 non-null  object
 24  AgeOfPolicyHolder     15420 non-null  object
 25  PoliceReportFiled     15420 non-null  object
 26  WitnessPresent        15420 non-null  object
 27  AgentType             15420 non-null  object
 28  NumberOfSuppliments   15420 non-null  object
 29  AddressChange_Claim   15420 non-null  object
 30  NumberOfCars          15420 non-null  object
 31  Year                  15420 non-null  int64
 32  BasePolicy            15420 non-null  object
```

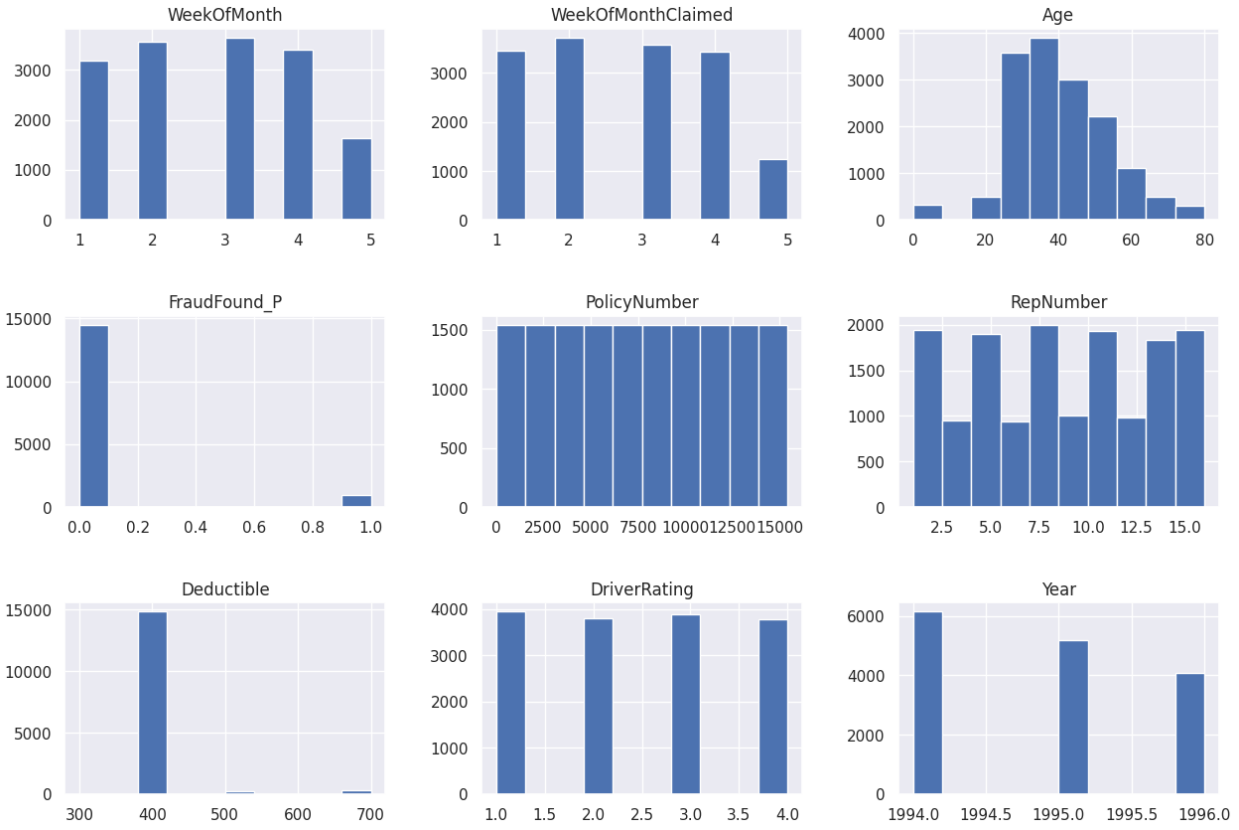
سپس با استفاده از تابع head() پنج نمونه ابتدایی این دیتاست را چک میشود و در کنار ان میتوان با استفاده از تابع describe() توضیحات دیگری مثل میانگین هر فیچر و ... را بدست اورد.

در مرحله بعدی که یکی از مراحل مهم در پیش‌پردازش داده ها میباشد باید دیتا هایی که بدون مقدار ان یا مقدار تعریف نشده (NaN) در ان وجود دارد پیدا و حذف شود. این کار با استفاده از تابع ()dopna انجام میشود که کافی است روی دیتافریم صدا زده شود تا این مقادیر بنابر سلیقه مثلا حذف شوند. حال برای چک کردن حذف این مقادیر از تمامی فیچر ها توابع ()is_null().sum روی دیتافریم صدا زده میشود تا تعداد مقادیر بدون مقدار یا تعریف نشده را در این دیتاست برای هر فیچر چاپ کند. این دیتاست به طور کلی از پیش اماده شده بود و هیچ مقدار تعریف نشده ای نداشت پس در نهایت برای هر فیچر خواهیم داشت:

```
Month                   0
WeekOfMonth             0
DayOfWeek               0
Make                    0
AccidentArea            0
DayOfWeekClaimed        0
MonthClaimed            0
WeekOfMonthClaimed      0
Sex                     0
MaritalStatus           0
Age                     0
Fault                   0
PolicyType              0
VehicleCategory         0
VehiclePrice            0
FraudFound_P            0
PolicyNumber            0
RepNumber               0
Deductible              0
DriverRating            0
Days_Policy_Accident    0
Days_Policy_Claim       0
PastNumberOfClaims      0
AgeOfVehicle            0
AgeOfPolicyHolder       0
PoliceReportFiled       0
WitnessPresent          0
AgentType               0
NumberOfSuppliments     0
AddressChange_Claim     0
NumberOfCars            0
Year                    0
BasePolicy              0
```

حال که دیتا اماده پردازش است سعی میشود با رسم شکل هایی ارتباط دیتاها باهم و یا ارزش هر فیچر و به طور کلی پراکندگی ای که هر فیچر دارد بیشتر اشنا شویم. این کار از انجایی مهم است که به دلیل تعدد فیچر ها مهم ترین های ان که بیشترین ارتباط را با چیزی که باید پیش‌بینی شود دارند یافت شوند.

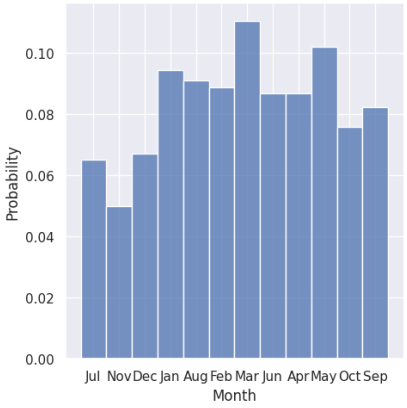
با فراخوانی تابع hist() روی دیتافریم میتوانیم نمودار پراکندگی هیستوگرام برای تمامی فیچر های عددی خود داشته باشیم تا ببینیم پراکندگی کدام یک از انها به صورت کاربردی رخ داده است.



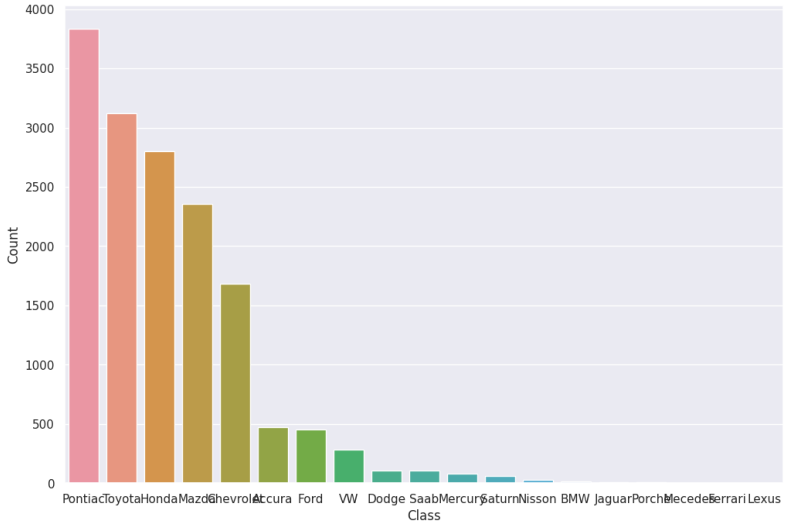
برای مثال میبینیم فیچر policy number پراکندگی مشخصی ندارد پس استفاده از این فیچر برای پیش‌بینی موردنظر کار درستی نخواهد بود. برعکس فیچر age پراکندگی مناسبی دارد که میتواند برای مدل مفید باشد.

سپس میتوانیم برای فیچر هایی که به نظر می اید تاثیری در پیش‌بینی داشته باشند نمودار جداگانه نیز رسم کنیم و به طور جداگانه انها را بررسی کنیم.

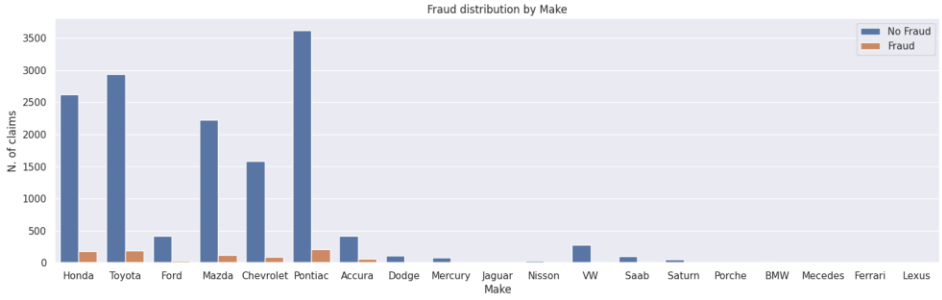
در این نمودار بررسی شده با چه احتمالی در هر ماه یک تصادف جعلی ثبت میشود.



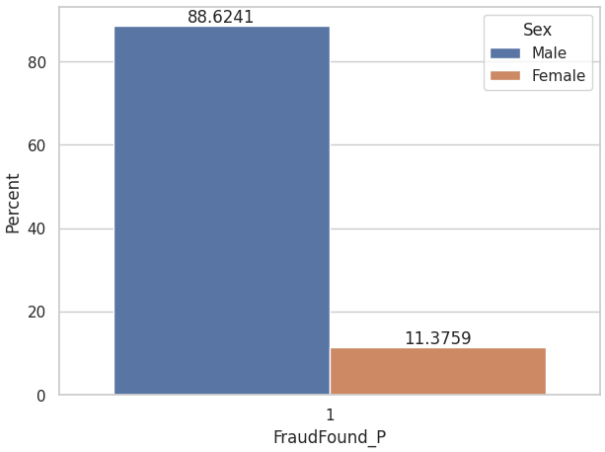
در این نمودار بررسی شده چه تعداد تصادف در هر نمونه ماشین رخ میدهد(تصادف جعلی و واقعی).



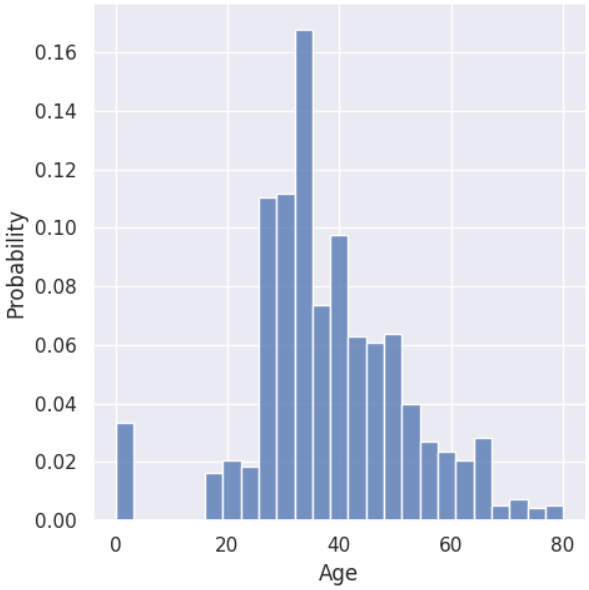
در این نمودار بررسی شده تصادف واقعی و جعلی به ازای هر مدل از ماشین ثبت میشود.



در این نمودار بررسی شده چه تعداد تصادف جعلی برای هر جنس مونث یا مذکر ثبت شده است.



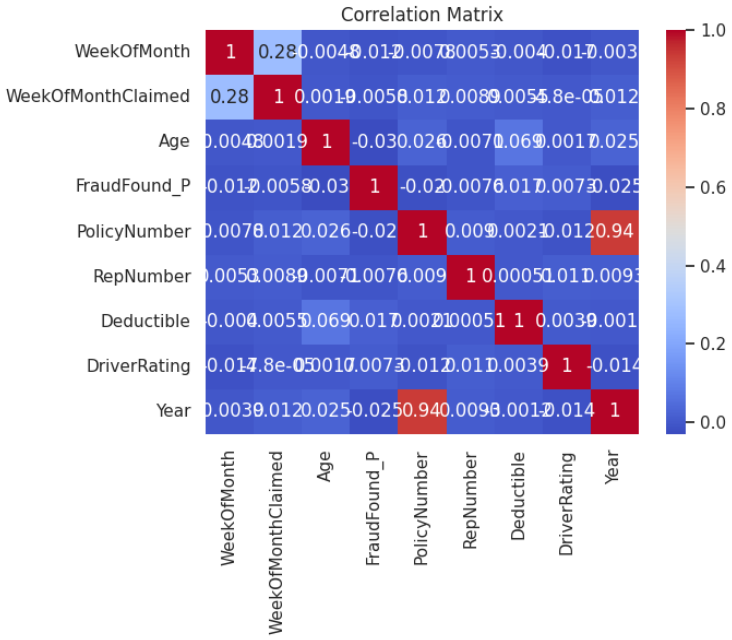
در این نمودار بررسی شده سن افرادی که تصادف جعلی ثبت کرده اند در چه رنجی به چه شکلی بوده است.



مثال های بیشتری که در بخش کد اورده شده است.

سپس میتوانیم ماتریس Correlation بین داده های عددی را رسم کنیم تا ارتباط هر یک باهم را نیز بررسی کنیم.

میبینیم برای مثال fraudFound_P ارتباط زیادی با فیچر Age دارد که در ادامه میتوانیم از ان استفاده کنیم.



از انجایی که فیچر های زیادی وجود دارد که مقادیر شان عددی نیست نیاز است تمامی انها یا به انتخاب انهایی را که نیاز داریم encode کنیم. برای این کار از کتاب‌خانه preprocessing.LabelEncode استفاده میکنیم. سپس ان را روی دیتافریم مان با تابع aplly() فراخوانی میکنیم. خواهیم دید که تمام لیبل های غیر عددی به صورت اینتیجر عددی در خواهند امد که مدل خواهد توانست روی ان کار کند.

در این بخش دیتا برای کار کردن مدل با ان امده است و میتوانیم با بخش کردن دیتا به تست و ترین از ان استفاده کنیم.

بخش دوم مسئله: امتحان کردن مدل های مختلف روی دیتاست (testing different models)

1- Logistic regression

این مدل با استفاده از کتاب خانه اماده اش یعنی LogisticReression اضافه شده و دیتای تست و ترین روی ان فیت میشود و نمونه هایی با ان پیش‌بینی میشود. خواهیم دید در این مدل به دقت تقریبا 0.936 درصد خواهیم رسید.

2- SVM

این مدل با استفاده از کتاب خانه اماده اش یعنی SVC اضافه شده و دیتای تست و ترین روی ان فیت میشود و نمونه هایی با ان پیش‌بینی میشود. خواهیم دید در این مدل نیز به دقت تقریبا 0.938 درصد خواهیم رسید.

3- Decision Tree

این مدل با استفاده از کتاب خانه اماده اش یعنی DecisionTreeClassifier اضافه شده و دیتای تست و ترین روی ان فیت میشود و نمونه هایی با ان پیش‌بینی میشود. خواهیم دید در این مدل نیز به دقت تقریبا 0.928 درصد خواهیم رسید.

4- Random Forest

این مدل با استفاده از کتاب خانه اماده اش یعنی RandomForestClassifier اضافه شده و دیتای تست و ترین روی ان فیت میشود و نمونه هایی با ان پیش‌بینی میشود. خواهیم دید در این مدل نیز به دقت تقریبا 0.937 درصد خواهیم رسید.

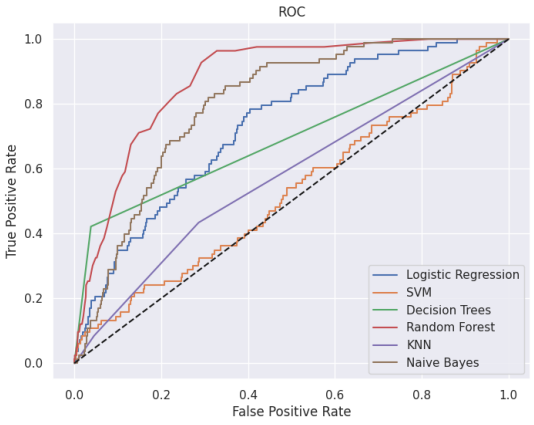
5- KNN

این مدل با استفاده از کتاب خانه اماده اش یعنی KNeighborsClassifier اضافه شده و دیتای تست و ترین روی ان فیت میشود و نمونه هایی با ان پیش‌بینی میشود. خواهیم دید در این مدل نیز به دقت تقریبا 0.936 درصد خواهیم رسید.

6- Naïve Bayes

این مدل با استفاده از کتاب خانه اماده اش یعنی GuassianNB اضافه شده و دیتای تست و ترین روی ان فیت میشود و نمونه هایی با ان پیش‌بینی میشود. خواهیم دید در این مدل نیز به دقت تقریبا 0.843 درصد خواهیم رسید.

برای مقایسه بهتر این مدل ها از نمودار roc استفاده میکنیم تا بهترین مدل ها بدست اید.



همان طور که دیده میشود مدل random forest عملکرد بهتری روی این دیتاست دارد چون دیده میشود سطح زیر نمودار (AUC) ان بیشتر است.

بخش سوم مسئله: استفاده از cross_validation (testing different models)

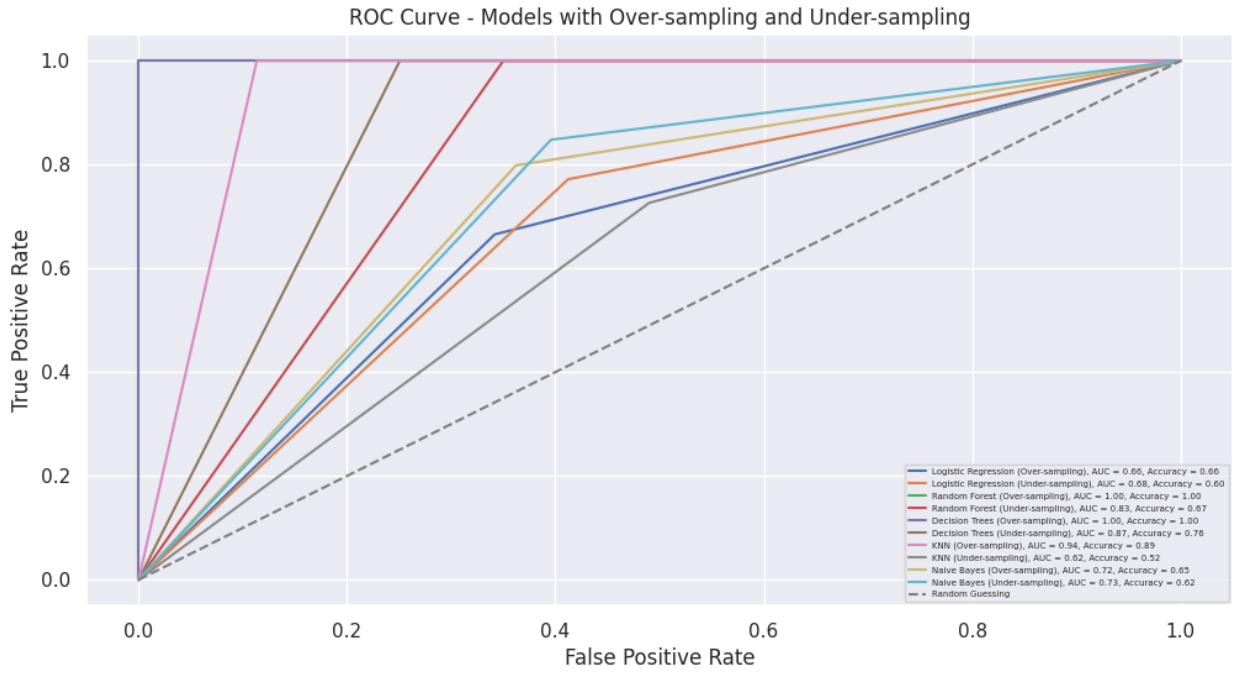
میدانیم که تکنیک های متفاوتی برای cross_validation وجود دارد مثل Hold_out یا K_fold و ... . ما در این تمرین از کتابخانه statifiedKFold استفاده کردیم تا بهترین عملکرد را برای مدل های موردنظر مان پیدا کنیم. برای اینکار مدل هایمان را لیست کردیم و هربار در دسته های جداگانه از دیتا دقت مدل را به دست اوردیم. در نهایت مقدار میانگینی که از تمامی دسته ها برای دقت ان مدل بدست امد را چاپ کردیم. این کار برای این انجام شد که از عملکرد واقعی و بررسی overfitting مطلع شویم.

```
Model: Logistic Regression
Cross-Validation Accuracy: [0.9342252  0.93526786 0.93415179 0.93526786 0.93415179]
Average Accuracy: 0.9346128961618092
----------------------------------------
Model: Decision Tree
Cross-Validation Accuracy: [0.91415831 0.94196429 0.91517857 0.92745536 0.89955357]
Average Accuracy: 0.9196620182353878
----------------------------------------
Model: Random Forest
Cross-Validation Accuracy: [0.93088071 0.93638393 0.93973214 0.93861607 0.93526786]
Average Accuracy: 0.9361761426978819
----------------------------------------
Model: K-Nearest Neighbors
Cross-Validation Accuracy: [0.92976589 0.93415179 0.93191964 0.93415179 0.93415179]
Average Accuracy: 0.9328281772575251
----------------------------------------
Model: Support Vector Machines
Cross-Validation Accuracy: [0.9342252  0.93526786 0.93526786 0.93526786 0.93415179]
Average Accuracy: 0.9348361104475235
----------------------------------------
Model: Naive Bayes
Cross-Validation Accuracy: [0.80602007 0.8046875  0.76227679 0.77232143 0.828125  ]
Average Accuracy: 0.7946861562350692
```

همان طور که دیدیم باز هم مدل های random forest , svm نتایج مناسبی داشتند و به نظر می ایند مدل خوبی برای این دیتاست میتوانند باشند.

بخش چهارم مسئله: چک کردن دیتای imbalanced

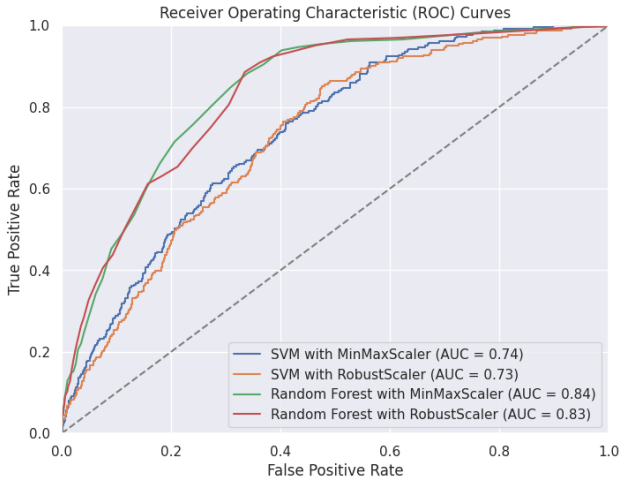
ما از تکنیک هایی مثل over_sampling و under_sampling برای هندل کردن دیتای دارای توزیع نامتعادل استفاده کردیم. در اصل عملکرد مدل را یک بار با استفاده از این تکنیک ها و بار دیگر بدون این تکنیم ها رسم کردیم. سپس نمودار roc برای زمانی که از این تکنیک ها استفاده کردیم را رسم کردیم. میبینیم که عملکرد تمامی مدل ها بسیار بهبود یافته و نشان میدهد توزیع های نامتعادل زیادی در دیتاست وجود داشته است.



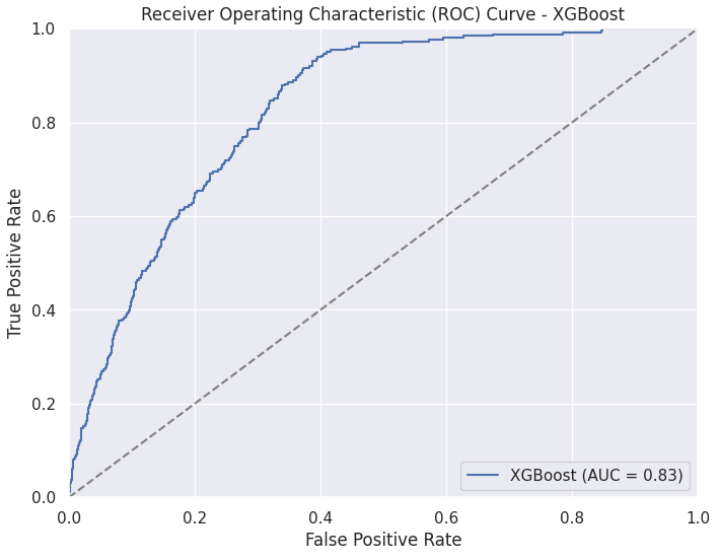
بخش پنجم مسئله: پیشرفت دادن مدل های SVM و random forest (boosting the performance of SVM and random forest models)

برای اینکار میتوانیم از تکنیک های مختلفی استفاده کنیم. برای مثال ما انتخاب کردیم که یکی سری پیش پردازش های اضافه تری نیز روی دیتاست انجام دهیم که عمکرد را بیشتر بهبود بخشد. برای این کار از توابع MinMaxScaler، RobustScaler استفاده میکنیم.

اگر نمودار ROC را رسم کنیم خواهیم دید هنگامی که از این توابع استفاده میکنیم سطح زیر نمودار برای مدل ها افزایش می یابد و همین نشان‌دهنده این است که عملکرد و دقت مدل بهتر شده است.



برای boosting مدل میتوان از کتابخانه XGBoost نیز استفاده کرد. در این حالت عملکرد مدل به بهترین حالت خود رسیده و بهترین دقت را به ما میدهد.



در نتیجه این تمرین راه پیداکردن بهترین مدل های Classification و کارکردن و بهبود بخشیدن عملکرد این مدل هارا بات متدهای گفته شده اموختیم. میتوانستیم در این تمرین از متد های feature engineering نیز استفاده کنیم تا حتی عملکرد باز هم پیشرفت یابد. همچینین نرمالیزیشن های دیگری نیز برای فیچر ها وجود داشت که میتوانستیم انهارا چک کنیم و بهترین انها را استفاده کنیم. اما به طور کلی میتوان گفت حال طریقه حل یک مسئله classification را میدانیم.