Fraud detection based on audit data using Machine learning

Salman Farshi
ID:1520162042
Salman.farshi@northsouth.edu

Introduction: Fraud is a critical issue worldwide. Firms that resort to the unfair practices without the fear of legal repercussion have a grievous consequence on the economy and individuals in the society. Auditing practices are responsible for fraud detection. Audit is defined as the process of examining the financial records of any business to corroborate that their financial statements are in compliance with the standard accounting laws and principles

Data Collection: We got the data from Kaggle, data is well prepared, there are only few missing data, otherwise everything is ok to fit in ml model.

Features:

```
In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 776 entries, 0 to 775
Data columns (total 27 columns):
Sector_score     776 non-null float64
LOCATION_ID      776 non-null object
PARA_A           776 non-null float64
Score_A          776 non-null float64
Risk_A           776 non-null float64
PARA_B           776 non-null float64
Score_B          776 non-null float64
Risk_B           776 non-null float64
TOTAL            776 non-null float64
numbers          776 non-null float64
Score_B.1        776 non-null float64
Risk_C           776 non-null float64
Money_Value      775 non-null float64
Score_MV         776 non-null float64
Risk_D           776 non-null float64
District_Loss    776 non-null int64
PROB             776 non-null float64
RiSk_E           776 non-null float64
History          776 non-null int64
Prob             776 non-null float64
Risk_F           776 non-null float64
Score            776 non-null float64
Inherent_Risk    776 non-null float64
CONTROL_RISK     776 non-null float64
Detection_Risk   776 non-null float64
Audit_Risk       776 non-null float64
Risk             776 non-null int64
dtypes: float64(23), int64(3), object(1)
memory usage: 163.8+ KB
```

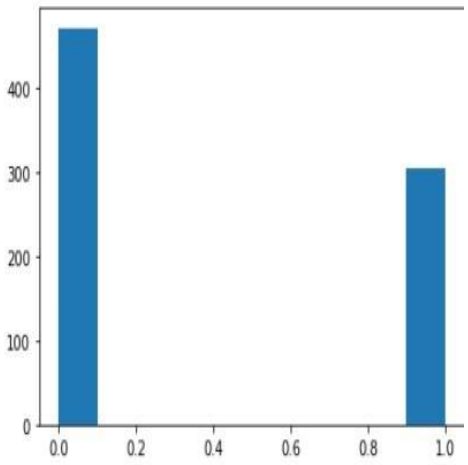
Graph:

In this graph we can see the correlation of the features. The most import features have highest value

```
In [8]: corr = df.corr()
        corr['Risk'].sort_values(ascending=False)

Out[8]: Risk            1.000000
        Score           0.785995
        Score_MV        0.688367
        Score_B         0.635768
        Score_A         0.619726
        CONTROL_RISK    0.416474
        RiSk_E          0.411803
        District_Loss   0.403806
        Risk_A          0.385067
        PARA_A          0.378758
        Inherent_Risk   0.357020
        Score_B.1       0.353803
        Risk_C          0.342140
        numbers         0.308141
        Prob            0.298639
        TOTAL           0.292022
        Money_Value     0.257097
        PARA_B          0.257029
        Risk_B          0.255286
        Risk_D          0.254355
        History         0.239453
        Audit_Risk      0.217113
        Risk_F          0.214511
        PROB            0.176912
        Sector_score   -0.394131
        Detection_Risk       NaN
        Name: Risk, dtype: float64
```

Target value distribution:

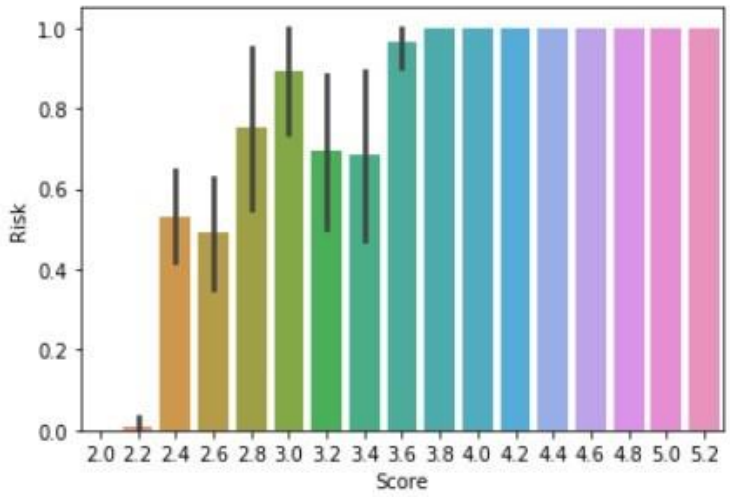
```
In [20]: plt.hist(x=df["Risk"])

Out[20]: (array([471.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 305.]),
          array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
          <a list of 10 Patch objects>)
```



Target vs some important features :

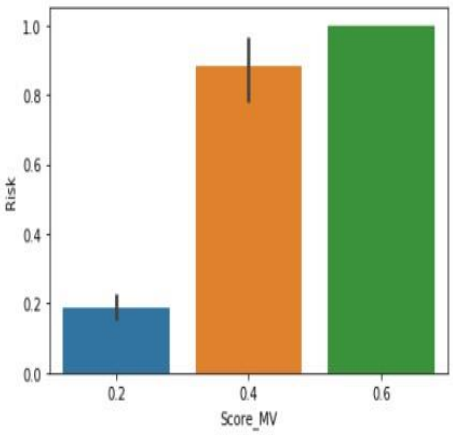
Score VS Risk

```
[23]: sns.barplot(df["Score"], y=df["Risk"])
[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffa0a266690>
```



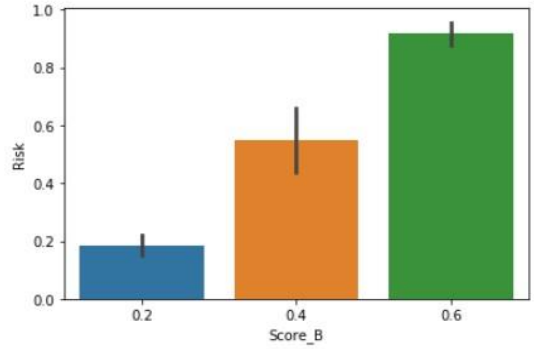
Score_MV VS Risk,

```
In [28]: sns.barplot(df["Score_MV"], y=df["Risk"])
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7f98809c7b10>
```

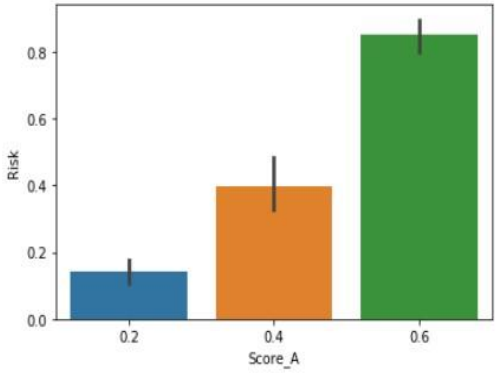


Others import features vs target

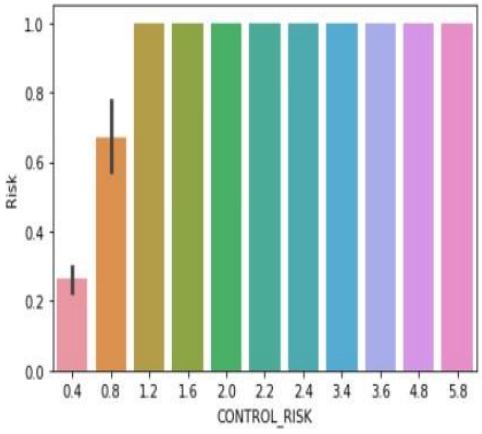
```
In [29]: sns.barplot(df["Score_B"], y=df["Risk"])
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9881023cd0>
```




```
In [30]: sns.barplot(df["Score_A"], y=df["Risk"])
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7f988142b050>
```




```
In [31]: sns.barplot(df["CONTROL_RISK"], y=df["Risk"])
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9880c85510>
```



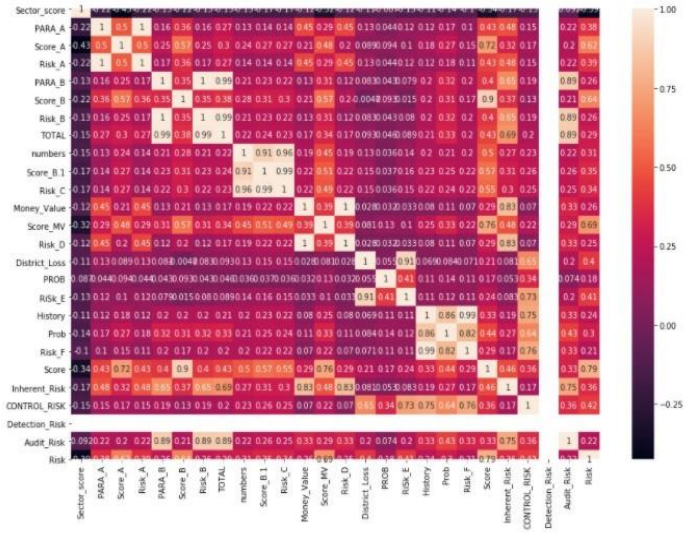
In the above figure we can observed that the data distribution and feature importance. When feature's values increase then risk increases

Correlation Matrix:

We have generated a good Correlation Matrix because its diagonal values is 1 . This types of correlation matrix give the most accurate prediction .

```
In [19]: plt.figure(figsize=(15,10))
         sns.heatmap(df.corr(),annot=True)
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffa1b943ed0>
```



Classifiers Use: SVM

Reasoning:

"**Support Vector Machine**" (**SVM**) is a supervised **machine** learning **algorithm** which can be used for both classification or regression challenges. The **SVM** classifier is a frontier which best segregates the two classes (hyper-plane/ line). In this project Sum works properly that Why I have used this algorithm.

Model :

In [14]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.35, random_state=42)
sc= StandardScaler()
scaled_X_train=sc.fit_transform(X_train)
scaled_X_test = sc.fit_transform(X_test)
from sklearn.svm import SVC
svc = SVC(class_weight='balanced')
from sklearn.model_selection import GridSearchCV
param_grid = {'C':[0.001,0.01,0.1,0.5,1],'gamma':['scale','auto']}
grid = GridSearchCV(svc,param_grid)
grid.fit(scaled_X_train,y_train)
```

```
/Users/salmanfarshi/opt/anaconda3/lib/python3.7/site-packages/sklearn/model_selectio
n/_split.py:1978: FutureWarning: The default value of cv will change from 3 to 5 in version
0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

Out[14]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=200, class_weight='balanced',
             coef0=0.0, decision_function_shape='ovr', degree=3,
             gamma='auto_deprecated', kernel='rbf', max_iter=-1,
             probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False),
       iid='warn', n_jobs=None,
       param_grid={'C': [0.001, 0.01, 0.1, 0.5, 1],
             'gamma': ['scale', 'auto']},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
       scoring=None, verbose=0)
```

Evolution Model: Training accuracy, sensitivity, precision, recall:

According to precision recall and f1 score we say that our model is working perfectly with 98% of accuracy. We have classified correctly all data expect 7+1 = 8 cases which is very acceptable for machine learning. For using grid search method, we found the best parameter for the data model without overfitting and under fitting.

```
In [15]: from sklearn.metrics import confusion_matrix,classification_report
         grid_pred = grid.predict(scaled_X_test)
         confusion_matrix(y_test,grid_pred)

Out[15]: array([[164,   7],
                [  1, 100]])
```

```
In [30]: print(classification_report(y_test,grid_pred))

                precision    recall  f1-score   support

             0       0.99      0.96      0.98       171
             1       0.93      0.99      0.96       101

      accuracy                           0.97       272
     macro avg       0.96      0.97      0.97       272
  weighted avg       0.97      0.97      0.97       272
```

ROC:

Due to some unknown update error I can not generate roc curve but I have tried a lot . This model is tested carefully and hopefully is has no overfitting issues

```
In [3]: from sklearn.metrics import plot_roc_curve
        plot_roc_curve(grid,y_test,grid_pred)

---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-3-30823e622d3c> in <module>
----> 1 from sklearn.metrics import plot_roc_curve

ImportError: cannot import name 'plot_roc_curve' from 'sklearn.metrics' (/Users/salmanf
arshi/opt/anaconda3/lib/python3.7/site-packages/sklearn/metrics/__init__.py)
```

Conclusion: SVM works very good in this problem and we have gained a good accuracy that is almost 98%. Due to some obstacle, we cannot generate Roc curve but later We will fix it. later we may improve our work if needed using others types of model so that we can compare the best model