Centre of Development of Advanced Computing (C-DAC), Pune

A Project Report on

"Corporate Credit Rating Forecasting"

Submitted in partial fulfilment for the award of
PG-Diploma in Big Data Analytics from C-DAC ACTS (Pune)

[March - 2022]

Presented By
Mr. Ajay Kumar (PRN 220340125005)
Mr. Aadesh Shinde (PRN 220340125043)
Mr. Aakash Shukla (PRN 220340125002)
Mr. Tushar Jadhav (PRN 220340125016)
Mr. Utsav Asthana (PRN 220340125056)

Guided By
Dr. Krishnanjan B

Centre of Development of Advanced Computing (C-DAC), Pune

CERTIFICATE

This is to certify that

Mr. Ajay Kumar	(PRN 220340125005)
Mr. Aadesh Shinde	(PRN 220340125043)
Mr. Aakash Shukla	(PRN 220340125002)
Mr. Tushar Jadhav	(PRN 220340125016)
Mr. Utsav Asthana	(PRN 220340125056)

have successfully completed their project on **"Corporate Credit Rating Forecasting"** under the guidance of **Dr. Krishnanjan B**, in partial fulfilment for the award of PG-Diploma in Big Data Analytics from C-DAC ACTS (Pune).

Project Guide
Dr. Krishnanjan B

Project Supervisor

HOD ACTS
Mr. Sundar Gaur

ACKNOWLEDGEMENT

This project '**Corporate Credit Rating Forecasting.**' was a great learning experience for us and we are submitting this work to Advanced Computing Training School (**CDAC ACTS**).

We all are very glad to mention the name of Dr. Krishnanjan B for his valuable guidance to work on this project. We cannot thank him enough for his patience, energy, and contagious positive attitude, and critical comments and for conscientious guidance and encouragement to accomplish this Project.

Our most heartfelt gratitude goes to Mr Sunder Gaur (HOD, ACTS), Ms. Risha P.R. (Program Head, ACTS) and Ms. Priyanka Ranade (Course Coordinator, PGDBDA) who gave us all the required support and coordinated with us to provide all the necessities we needed tocomplete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

We also acknowledge with a deep sense of reverence, our gratitude towards our parents and member of our family, who has always supported us morally. Last, but not least, gratitude goes to all our project team members who helped us complete this report. Any omission in this brief acknowledgement does not mean a lack of appreciation.

Thanking You.

Table of Contents

Corporate Credit Rating Forecasting.

Table of Figures

1. Abstract

Big data analytics are used primarily in various sectors for accurate prediction and analysis of the large data sets. They allow the discovery of significant information from large data sets, otherwise, it is hidden in plain sight.

Corporate credit ratings are comprehensive indicators of a company's management performance, earnings quality, and future prospects. They represent its market evaluation and status in the industry and are relevant to the financing and investment decision-making process. Financial institutions determine corporate credit ratings using corporate financial and governance indicators.

This study develops an approach to forecasting corporate credit ratings by analyzing past ratings from various rating agencies to assist investors in effectively evaluating and controlling investment risk. This objective is achieved through the following steps:

1. Designing a corporate credit rating forecasting process, scraping data from different websites, cleaning and imputing the data with the use of plots.
2. Developing techniques for corporate credit rating forecasting
3. Implementing and evaluating the corporate credit rating forecasting mechanism.

2. Introduction and Overview of Project

A Corporate credit rating is an assessment of the creditworthiness of a Investment company, often a investor's status with respect to a particular investment made in a corporate company. This assessment is based on comprehensive, defined rating methodology, and rating criteria. Credit ratings are judgments of investor based on relevant risk factors, expressed by a letter-grade or number-grade rating symbol that markets depend on for differentiating between different risk levels of a company.

Credit ratings have become one of the primary references for investors to assess and reduce possible risks, and make on-point decisions before investing in a company.

Data Scraping (Python- Beautiful Soup), Fetching data from a database, statistical analysis and machine-learning models on PySpark, have been deployed to **evaluate corporate credit risk** in this project.

3. Data Description and Technologies Implemented

3.1. Data Description

Below are the data Sources used for extracting the data:

3.1.1. Corpdatahub for data extraction.

Data present on this source was in tabular format and hence we need to scrap tables from this data source.

There is a separate file for company data and rating history from various agencies. So there in total 2 files after the successful gathering having 7k+ and 400k+ records respectively.

Also each file contains below mentioned information :

- **Rating history** : Rating agency, Company name, Date of Rating, Ratings (past ratings)
- **Company data** : Name, Symbol, Sector, Industry

3.1.2. Financial modeling prep API for Financial ratios data extraction.

Data gathered from this source is received in “json” format and stored in pandas dataframe which was then ingested to mongoDb database.

In total we gathered a single file having below mentioned information :

- **Liquidity measurement ratios**
- **Profitability indicator ratios**
- **Debt ratios**
- **Operating performance ratios**
- **Cash flow indicator ratios**
- **Investment Valuation ratios**

3.2. Information on the technologies being used:

3.2.1. Python:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

3.2.2. Beautiful Soup library of python:

Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

3.2.3. MongoDB Database:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License which is deemed non-free by several distributions.

3.2.4. PySpark:

PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing your data in a distributed environment. PySpark supports most of Spark's features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core.

3.2.5. Pandas DataFrame of python:

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

3.2.6. Scikit-Learn - Machine Learning library:

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines

3.2.7. WordCloud python technique:

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud.

3.2.8. Use Streamlit library:

We are using Streamlit for Visualization.

4. Initial Steps:

- We define our problem statement and accordingly we came to the conclusion on what type of data we require to achieve our target.
- The initial steps involve the research work to gain the data having company names for our prototype
- Then we began to look for various open-sources from where we can get the historical data for numerous stock companies .
- After sufficient research we fixated on 2 sources for our data, namely-

a) Corpdatahub for data extraction:

During our research for company data and their rating by various agencies we came across different sources but the data we got was not relevant, finally we found out this website Corpdatahub where we found out the right data for our project and we scrapped relevant data from the website.

b) Financial modeling prep API for Financial ratios data extraction:

For procuring the financial indicators of a company we got this api from the above mentioned website, through which we scrapped the financial ratios data and cinverted it into a dataframe and then stored in mongoDb for further use

5. Workflow

1. Problem Statement
2. Deciding on program structure
3. Setting & running of required Environments & Applications
4. Data Scraping
5. Data ingestion to database (mongoDb)
6. Exploratory data analysis (EDA) using PySpark
7. Development of a machine learning models
8. Hyperparameter tuning
9. Preparing Confusion matrix and feature selection
10. Visualize companies and making a suitable plot
11. Design UI for prediction using Streamlit

5.1. Problem Statement

a) Analysis or Prediction

As we all know, the risk of investment in a corporate company cannot be predicted. Hence the decision of doing predictive analysis was made based on pattern recognition from historical data.

b) Setting up Goal:

We settled down on a goal such that to shortlist the companies based on their risk.

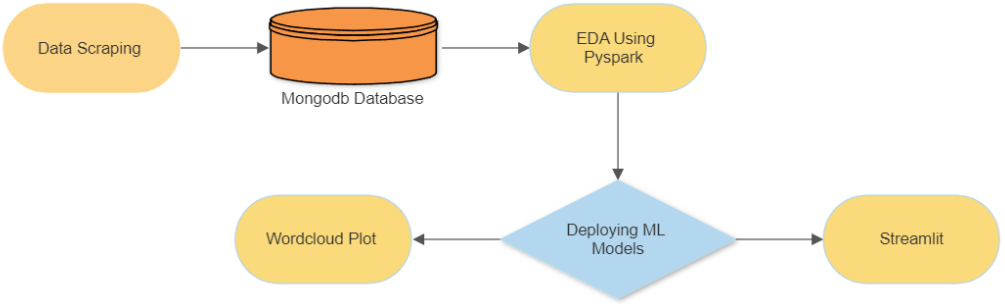
5.2. Deciding on program structure

a) Required programming languages:

Programming language to program the application was finalised as python (for performing actions).

b) Structure:

Program flow structure was worked upon and decided for ETL structure.
Refer the below file structure used for the programming :



5.3. Setting & running of required Environments & Applications

5.3.1. Environments:

1. Installing BeautifulSoup library:

Installing the bs4 library of python for pulling out data from HTML and XML files

1) Creating mongoDb database:

Creating a mongoDb database to stored all the scraped data.

2) Setting up mongoDb cluster:

Setting up mongoDB compatible cluster to retrieve dat from database.

3) PySpark jupyter notebook:

Using mongo-spark connector to fetch data from database and then using ML models and visualizations in a single notebook of PySpark

5.4. Data Scraping

Execution of project from data scraping

5.4.1. Corpdatahub for data extraction:

Data extraction from the above website included the use of BeautifulSoup library of python.

scraping Various Companies of US STOCKS

```
In [2]: url = 'https://financialmodling.netlify.app/data/us_stocks'
        url_contents = requests.get(url)
        soup = BeautifulSoup(url_contents.text, "html.parser")

In [3]: soup
<td>Agilent Technologies, Inc.</td>
<td>Capital Goods</td>
<td>Biotechnology: Laboratory Analytical Instruments</td>
</tr>
<tr>
<th>1</th>
<td>AA</td>
<td>Alcoa Corporation</td>
<td>Basic Industries</td>
<td>Aluminum</td>
</tr>
<tr>
<th>2</th>
<td>AACG</td>
<td>ATA Creativity Global</td>
<td>Consumer Services</td>
<td>Other Consumer Services</td>
</tr>
<tr>

In [4]: div = soup.find("thead")

        headers = []
        for i in div.find_all('th'):
            title = i.text
            headers.append(title)
        headers
Out[4]: ['', 'Symbol', 'Name', 'Sector', 'industry']

In [27]: client = pymongo.MongoClient("mongodb+srv://project9:project9@cluster0.5xyb5sn.mongodb.net/?retryWrites=true&w=majority")

         mydb=client['new']

         scraped_data = mydb.us_stocks

         Schema = {
             'Symbol': {
```

Fig. 1: Scraping data(1)

The code snippet of the scraping technique goes as follows:

```
In [7]: div = soup.find("tbody")
        div
Out[7]: <tbody>
        <tr>
        <th>0</th>
        <td>A</td>
        <td>Agilent Technologies, Inc.</td>
        <td>Capital Goods</td>
        <td>Biotechnology: Laboratory Analytical Instruments</td>
        </tr>
        <tr>
        <th>1</th>
        <td>AA</td>
        <td>Alcoa Corporation</td>
        <td>Basic Industries</td>
        <td>Aluminum</td>
        </tr>
        <tr>
        <th>2</th>
        <td>AACG</td>
        <td>ATA Creativity Global</td>

In [30]: row = div.find('tr')
         row
Out[30]: <tr>
         <th>0</th>
         <td>A</td>
         <td>Agilent Technologies, Inc.</td>
         <td>Capital Goods</td>
         <td>Biotechnology: Laboratory Analytical Instruments</td>
         </tr>

In [32]: row_data = [td.text.strip() for td in row]
         print(row_data)
         dict_data = {'Symbol':row_data[3], 'Name':row_data[5], 'Sector':row_data[7], 'industry':row_data[9]}
         dict_data
         #scraped_data.insert_one(dict_data)
```

Fig. 2: Scraping data(2)

```
In [32]: row_data = [td.text.strip() for td in row]
         print(row_data)
         dict_data = {'Symbol':row_data[3], 'Name':row_data[5], 'Sector':row_data[7], 'industry':row_data[9]}
         dict_data
         #scraped_data.insert_one(dict_data)

['', '0', '', 'A', '', 'Agilent Technologies, Inc.', '', 'Capital Goods', '', 'Biotechnology: Laboratory Analytical Instruments', '']

Out[32]: {'Symbol': 'A',
          'Name': 'Agilent Technologies, Inc.',
          'Sector': 'Capital Goods',
          'industry': 'Biotechnology: Laboratory Analytical Instruments'}

In [33]: count=0
         for row in div.find_all('tr')[1:]:
             row_data = [td.text.strip() for td in row]
             dict_data = {'Symbol':row_data[3], 'Name':row_data[5], 'Sector':row_data[7], 'industry':row_data[9]}
             scraped_data.insert_one(dict_data)
             count = count+1
             if count%500==0:
                 print(count)

500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
```

Fig. 3: Scraping data(3)

5.4.2. Financial modeling prep API for financial ratios data extraction:

Extraction of company data was performed using selenium python API from screener.in website. The website provides user functionality to download the performance indices of the company over the preceding 10 years in excel format.

```
In [24]: financial_ratios_df = pd.DataFrame()
         financial_ratios_df.head()

Out[24]: _

In [25]: import json
         from urllib.request import urlopen

         # Get every company from the list the financial information
         for Symbol in Symbol_list[880:]:
             if Symbol.isalpha() == True:

                 ratios = pd.DataFrame.from_dict(
                         get_jsonparsed_data("https://financialmodelingprep.com/api/v3/ratios/"
                                             + Symbol +
                                             "?limit=40&apikey=ea9f74285ac2407d723cf1356da4d9be"))

                 frames = [financial_ratios_df, ratios]

                 financial_ratios_df = pd.concat(frames)

In [26]: #financial_ratios_df.to_csv(r'C:\Users\AJAY\Desktop\Project\final\Financial_Ratios_Data\financial_ratios_df.csv')
         #financial_ratios_df_all_from_api = financial_ratios_df

In [27]: ########## Merging Daywise Data
         financial_ratios_df = pd.read_csv(r'C:\Users\AJAY\Desktop\Project\final\Financial_Ratios_Data\financial_ratios_df.csv')
         financial_ratios_df_all_from_api = financial_ratios_df.drop(['Unnamed: 0'] , axis=1)
```

Fig. 4: Financial Model API

5.5. Data ingesting to database (mongoDB)

Code Block:

```
client = pymongo.MongoClient("mongodb+srv://project9:project9@cluster0.5xyb5sn.mongodb.net/?retryWrites=true&w=majority")

mydb=client['new']

scraped_data = mydb.rating_history

Schema = {
    'Rating Agency Name': {
        'type': 'string',
        'minlength': 0,
        'required': True
    },
    'Name': {
        'type': 'string',
        'minlength': 0,
        'required': True
    },
    'Rating Action Date': {
        'type': 'string',
        "required": True,
        'minlength': 0
    },
    'Rating': {
        'type': 'string',
        'minlength': 0,
        'required': True
    }
}

scraped_data.insert_one( Schema)
```

Fig. 5: Py-Mongo Connection

5.6. Data retrieval using mongo-spark connector

Code block :

The retrieval of dataset for EDA and modeling is done using mongo-spark connector and the code goes as follows –

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

working_directory = '/home/talentum/spark/jars/*'
```

Here we are creating Spark connection to mongoDB.

```
database="new"
collection="corparate_credit_rating_datset"
connectionstring="mongodb+srv://project9:project9@cluster0.5xyb5sn.mongodb.net/?retryWrites=true&w=majority"
```

```
spark =SparkSession.builder.config('spark.mongodb.input.uri',connectionstring).config('spark.mongodb.output.uri',connectionstring).config('spark.jars.packages','org.mongodb.spark:mongo-spark-c
```

```
df = spark.read.format("com.mongodb.spark.sql.DefaultSource").option("uri",connectionstring).option('database',database).option('collection',collection).load()
```

Fig. 6: Mongo-Spark Connection

6. Analysis

6.1. Exploratory Data analysis (EDA) using PySpark

After successfully gathering, ingesting and retrieval of data here comes the turn of EDA. The actions performed during EDA were- **viewing the dimensions, viewing the structure, analyse rating labels, descriptive statistics, skewness and outliers, normalize the data, plots.**

```
from pyspark.sql.functions import skewness

for i in df3.columns[6:-1]:
    print(f'Column Name : {i}')
    df3.agg(skewness("operatingProfitMargin")).show()
    print()
```

```
Column Name : assetTurnover
+--------------------------------+
|skewness(operatingProfitMargin)|
+--------------------------------+
|             -40.917073713020194|
+--------------------------------+
```

Fig. 7: EDA(Skewness)

```
[ ] for c in df_rating.columns[6:-1]:

        q1 = df_rating[c].quantile(0.25)
        q3 = df_rating[c].quantile(0.75)
        iqr = q3 - q1 #Interquartile range
        fence_low  = q3-1.5*iqr
        fence_high = q1+1.5*iqr
        lower_out = len(df_rating.loc[(df_rating[c] < fence_low)  ,c])
        upper_out = len(df_rating.loc[(df_rating[c] > fence_high)  ,c])
        outlier_count = upper_out+lower_out
        prop_out = outlier_count/len(df_rating)
        print(c, ": "+"{:.2%}".format(prop_out))
```

```
assetTurnover : 15.35%
cashFlowToDebtRatio : 21.17%
cashPerShare : 15.89%
cashRatio : 16.39%
currentRatio : 18.31%
daysOfSalesOutstanding : 22.51%
debtRatio : 17.62%
ebitPerRevenue : 23.20%
effectiveTaxRate : 22.70%
enterpriseValueMultiple : 23.74%
freeCashFlowOperatingCashFlowRatio : 18.95%
freeCashFlowPerShare : 24.83%
grossProfitMargin : 13.23%
interestCoverage : 22.70%
netProfitMargin : 27.69%
operatingCashFlowPerShare : 18.26%
operatingCashFlowSalesRatio : 17.52%
```

Fig. 8: EDA (stat)

```
df_rating_outlier = df_rating.copy()

for c in df_rating_outlier.columns[5:30]:

    q1 = df_rating_outlier[c].quantile(0.25)
    q3 = df_rating_outlier[c].quantile(0.75)
    iqr = q3 - q1 #Interquartile range
    fence_low  = q3-1.5*iqr
    fence_high = q1+1.5*iqr

    for i in range(len(df_rating_outlier)):

        if df_rating.loc[i,c] < fence_low or df_rating.loc[i,c] > fence_high: # if Outlier

            df_rating_outlier.loc[i,c] = 1

        else: # Not Outlier
            df_rating_outlier.loc[i,c] = 0

df_rating_outlier["total"] = df_rating_outlier.sum(axis=1)
df_rating_outlier.total.hist(bins = 10)
plt.show()
```

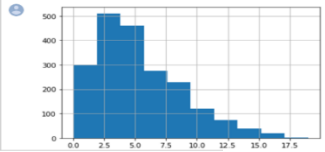


Fig. 9: EDA (Outlier)

6.2. Deployment of machine learning models

In the following steps we will perform the following:

1. Prepare the dataset and Split in train and test:

We have to prepare dataset for analysis by encoding, have to decide response variable and have to divide it in Train Test Split.

```
[ ] df_rating=df_rating.drop(['Rating'],axis=1)
    df_rating['Rating']=df_rating['Rating desc']
    df_rating=df_rating.drop(['Rating desc'],axis=1)

[ ] le = preprocessing.LabelEncoder()
    le.fit(df_rating.Sector)
    df_rating.Sector = le.transform(df_rating.Sector) # encode sector
    le.fit(df_rating['Rating'])
    df_rating['Rating'] = le.transform(df_rating['Rating']) # encode rating

[ ] df_rating= df_rating.dropna()

[ ] df_train, df_test = train_test_split(df_rating, test_size=0.2, random_state = 2022)

[ ] X_train = df_train.drop(['Clean_Name','Rating','Rating Agency Name','Symbol'], axis=1)
    y_train = df_train.Rating
    X_test  = df_test.drop(['Clean_Name','Rating','Rating Agency Name','Symbol'] , axis=1)
    y_test  = df_test.Rating
```

Fig. 10: Prepare dataset

2. Test a wide range of ML models (Tree-based, Probabilistic and so on).
We have applied multiple Regression Algorithms on our dataset to test the accuracy. The models consist of:

- **XGBoost**

```
[ ] XGB_model = xgb.XGBRegressor(objective ='multi:softmax', num_class =4)
    XGB_model.fit(X_train, y_train)
    y_pred_XGB = XGB_model.predict(X_test)
    Accuracy_XGB = metrics.accuracy_score(y_test, y_pred_XGB)
    print("XGB Accuracy:",Accuracy_XGB)

    [03:41:13] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the
    XGB Accuracy: 0.645320197044335
```

Fig. 11: XGBoost

- **Random Forest**

```
[ ]  RF_model = RandomForestClassifier(random_state=1234)
     RF_model.fit(X_train,y_train)
     y_pred_RF = RF_model.predict(X_test)
     Accuracy_RF = metrics.accuracy_score(y_test, y_pred_RF)
     print("RF Accuracy:",Accuracy_RF)

     RF Accuracy: 0.6182266009852216
```

Fig. 12: Random Forest

- **Gradient Boosting**

```
[ ]  GBT_model = GradientBoostingClassifier(random_state=123)
     GBT_model.fit(X_train, y_train)
     y_pred_GBT = GBT_model.predict(X_test)
     Accuracy_GBT = metrics.accuracy_score(y_test, y_pred_GBT)
     print("GBT Accuracy:",Accuracy_GBT)

     GBT Accuracy: 0.6133004926108374
```

Fig. 13: Gradient Boosting

- **Support Vector Machine (SVM)**

```
[ ]  SVC_model = svm.SVC(kernel='linear', gamma= 2, C = 5, random_state=1234)
     SVC_model.fit(X_train, y_train)
     y_pred_SVM = SVC_model.predict(X_test)
     Accuracy_SVM = metrics.accuracy_score(y_test, y_pred_SVM)
     print("SVM Accuracy:",Accuracy_SVM)

     SVM Accuracy: 0.5098522167487685
```

Fig. 14: Support Vector Machine

- **Logistic Regression**

```
[ ]  LR_model = LogisticRegression(random_state=1234 , multi_class='multinomial', solver='newton-cg')
     LR_model = LR_model.fit(X_train, y_train)
     y_pred_LR = LR_model.predict(X_test)
     Accuracy_LR = metrics.accuracy_score(y_test, y_pred_LR)
     print("LR Accuracy:",Accuracy_LR)

     LR Accuracy: 0.49014778325123154
```

Fig. 15: Logistic Regression

- **Linear Discriminant Analysis (LDA)**

```
[ ]  LDA_model = LinearDiscriminantAnalysis()
     LDA_model.fit(X_train,y_train)
     y_pred_LDA = LDA_model.predict(X_test)
     Accuracy_LDA = metrics.accuracy_score(y_test, y_pred_LDA)
     print("LDA Accuracy:",Accuracy_LDA)

     LDA Accuracy: 0.4852216748768473
```

Fig. 16: Linear Discriminant Analysis

- **KNN**

```
[ ]  KNN_model = KNeighborsClassifier(n_neighbors = 3)
     KNN_model.fit(X_train,y_train)
     y_pred_KNN = KNN_model.predict(X_test)
     Accuracy_KNN = metrics.accuracy_score(y_test, y_pred_KNN)
     print("KNN Accuracy:",Accuracy_KNN)

     KNN Accuracy: 0.4753694581280788
```

Fig. 17: KNN

- **Neural Network**

```
MLP_model = MLPClassifier(hidden_layer_sizes=(5,5,5), activation='logistic', solver='adam', max_iter=1500)
MLP_model.fit(X_train, y_train)
y_pred_MLP = MLP_model.predict(X_test)
Accuracy_MLP = metrics.accuracy_score(y_test, y_pred_MLP)
print("MLP Accuracy:",Accuracy_MLP)

MLP Accuracy: 0.3694581280788177
```

Fig. 18: Neural Network

- **Quadratic Discriminant Analysis (QDA)**

```
[ ]  QDA_model = QuadraticDiscriminantAnalysis()
     QDA_model.fit(X_train,y_train)
     y_pred_QDA = QDA_model.predict(X_test)
     Accuracy_QDA = metrics.accuracy_score(y_test, y_pred_QDA)
     print("QDA Accuracy:",Accuracy_QDA)

     QDA Accuracy: 0.21921182266009853
```

Fig. 19: Quadratic Discriminant Analysis

- **Naïve Bayes**

```
[ ]  GNB_model = GaussianNB()
     GNB_model.fit(X_train, y_train)
     y_pred_GNB = GNB_model.predict(X_test)
     Accuracy_GNB = metrics.accuracy_score(y_test, y_pred_GNB)
     print("GNB Accuracy:",Accuracy_GNB)

     GNB Accuracy: 0.04926108374384237
```

Fig. 20: Naïve Bayes

3. Compare the accuracry of all models

From testing the different algorithms we have compared accuracy for them as follows:

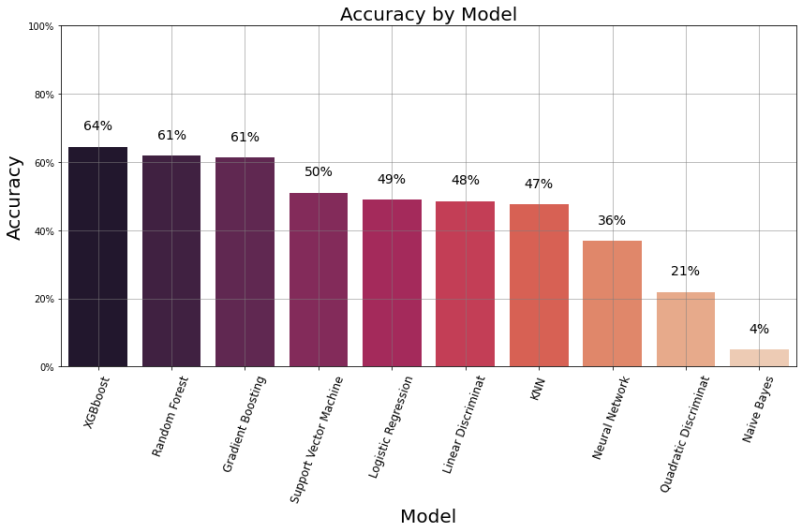


Fig. 21: Accuracy Comparison

6.3. Hyperparameter tuning

Now we tried to increase the performance even more. We will use a cross-validation approach

Choose our winning model and tune hyperparameters to target a higher accuracy. First we load the train and test data into DMatrices. `DMatrix` is a data structure used by XGBoost to optimize both memory efficiency and training speed.

DMatrices. DMatrix is a data structure used by XGBoost to optimize both memory efficiency and training speed.

```
[ ]  dtrain = xgb.DMatrix(X_train, label=y_train)
     dtest = xgb.DMatrix(X_test, label=y_test)
```

The params dictionary

We create a dictonary with the parameters from our previous XGboost model.

```
[ ]  params = XGB_model.get_xgb_params()
```

```
[ ]  params['eval_metric'] = "merror"
     num_boost_round = 1000
     params['max_depth'] = 13
     params['min_child_weight'] = 7
```

The num_boost_round which corresponds to the maximum number of boosting rounds that we allow.

Results

This are the final parameters of our tunned model.

```
[ ]  model = xgb.train(
         params,
         dtrain,
         num_boost_round=num_boost_round,
         evals=[(dtest, "Test")],
         early_stopping_rounds=1000,
         verbose_eval=100
     )
```

Fig. 22: Hyperparameter Tuning

6.4. Preparing Confusion matrix and feature selection using PySpark

6.4.1. Confusion matrix

We will now analyse according to each class the performance of the model. The best way to do it is with a confusion matrix. We can see how many points were missclassified and where were then classified to if not the right rating.

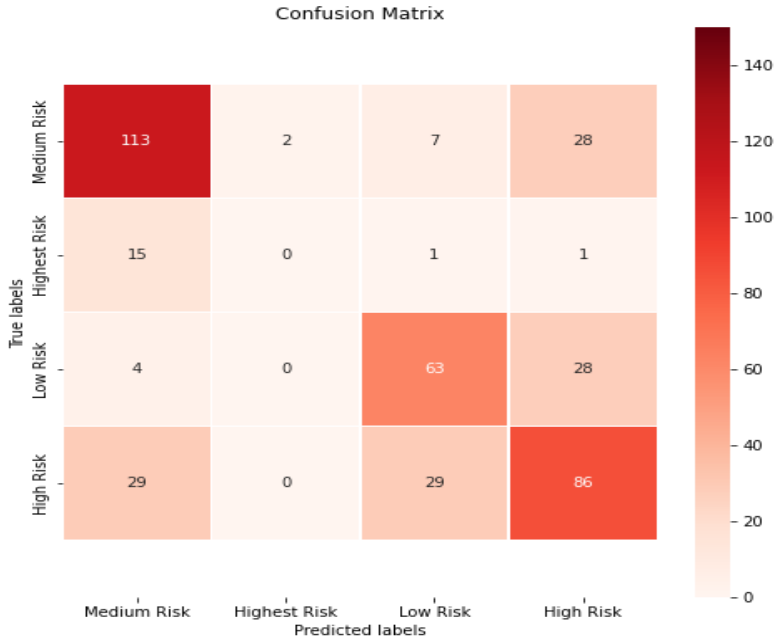


Fig. 23: Confusion Matrix

6.4.2. Feature selection

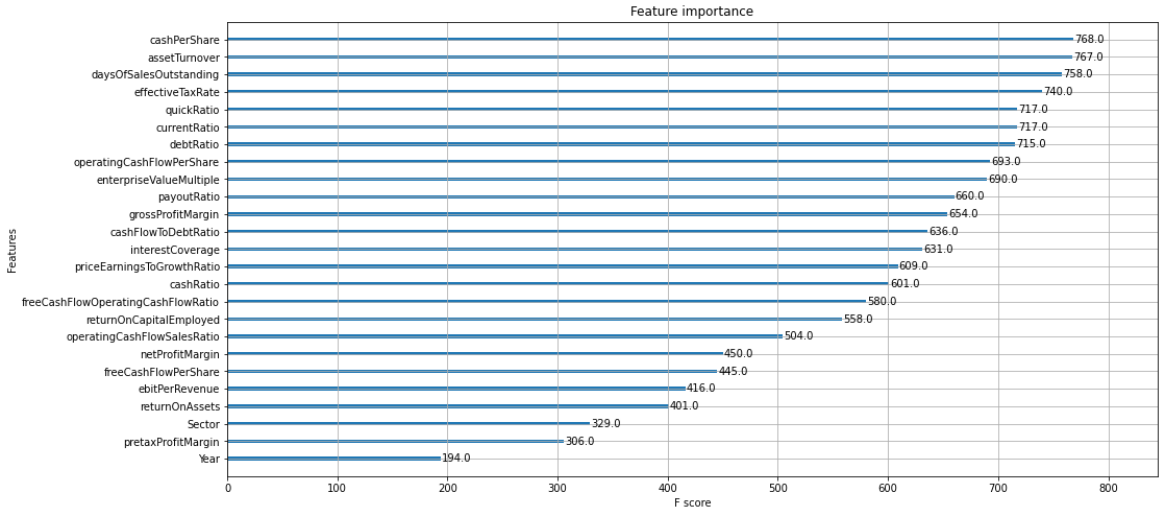


Fig. 24: Feature Selection

In our tast task we will identify which features were the most valuable for our model. In our first step we will check if by any chance we can increase the accuracy of our model extracting a feature.

6.5. Visualize companies and making a suitable plot using WordCloud

In this step we are going to visualize the company using the wordCloud technique of python in PySpark.

Fig. 25: WordCloud

6.6. Design UI for Prediction:

We have used well designed UI for prediction using Stremlit library, where we have to provide company symbol and have to choose company sector from dropdown list as input variable which will give risk prediction for given company for five years.

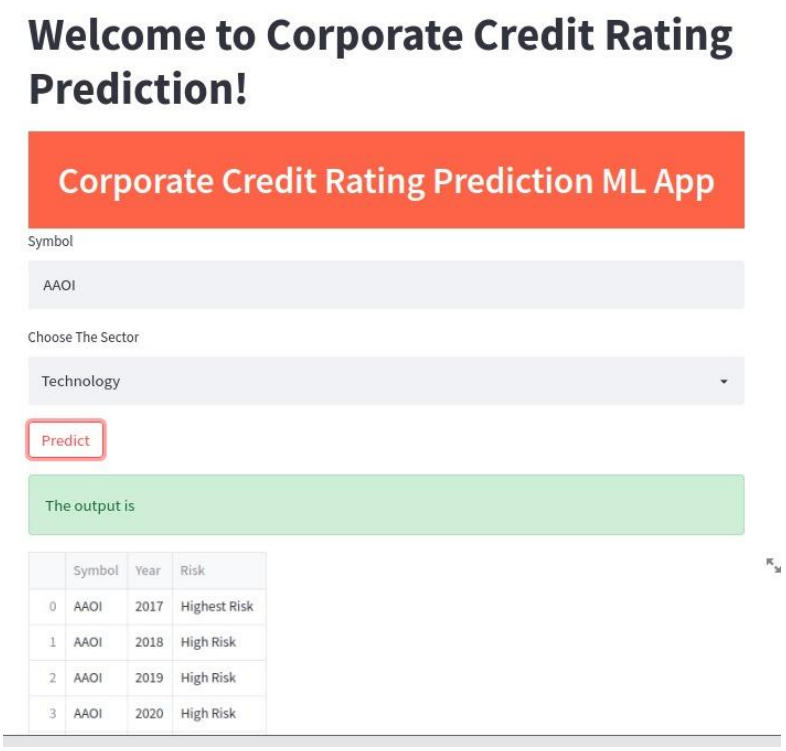


Fig. 26: Visualization

7. Project Timeline:

Since time is very important we have to plan everything. We had prepared plan for that and have performed works accordingly.

Timeline:

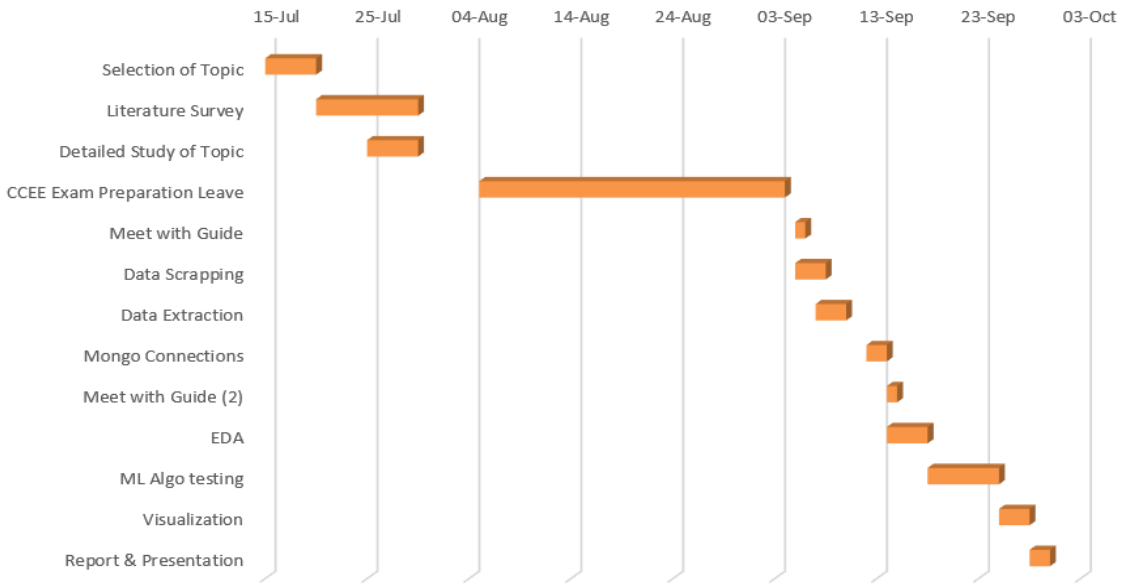


Fig. 27: Timeline

8. Conclusion

In this Project, we have Successfully implemented various techniques like beautiful soup for Data scrapping, MongoDB as database, PySpark for EDA, Machine Learning Models are used for prediction and efficient Corporate Credit rating Forecasting also we have displayed output of the project using Streamlit library. Credit ratings play a central role in the loaning decisions of financial institutions. In addition, corporations refer to the credit ratings of their trading partners in business trade. However, financial institutions and corporations must spend money to acquire credit rating indices. Moreover, credit rating indices are not released promptly by credit rating agencies. As a result, financial institutions such as banks establish risk management departments to develop credit rating prediction models used to obtain immediate credit information. Corporations also suffer from trading risk because of bad debts from the accounts and notes they receive.

This study proposes an effective credit rating prediction model to financial institutions and corporations as well as to assist investors in effectively evaluating and controlling investment risk.

9. Future Scope

- For increasing the accuracy of the code, ML-Lib can also be included in the processof making predictions.
- The user Interface can be improved according to our needs for better visualization as well as to include more parameters.
- The Accessibility of project can be improved more by uploading project on Docker image or can be deployed on Cloud.
- The Predictions can be improved by reducing existing or including new Features in dataset.
- It can be used precisely to invest in companies by increasing accuracy score.

10.Bibliography:

Website for Scraping: https://corpdatahub.netlify.app/

API Link: https://site.financialmodelingprep.com/developer

Apache Spark Reference: https://spark.apache.org/docs/latest/

Pyspark Documentation: https://spark.apache.org/docs/latest/api/python/

Beautiful Soup Documentation: https://beautiful-soup-4.readthedocs.io

Streamlit Documentation: https://docs.streamlit.io/library/