- **APPROACH**

  I used three classifiers which were Decision Tree, SVM & XG Boost. I compared the F1 Score of these models and finally selected XG boost as the best model for Credit Risk Prediction.

- **PRE-PROCESSING**

- To deal with any missing data I used **SimpleImputer** module of sklearn. Simple Imputer was applied to attributes from F1-F9.
- I applied **OneHotEncoder** to the race column(F10) in the dataset, so that it could be easier for machine to learn from the data.
- For sex column(F11), I used **LabelEncoder.** So, the male was converted to 1 and female to 0.
- I dropped the ID column since it was not useful for any prediction.

```
In [32]: from sklearn.impute import SimpleImputer                          #dealing with missing data
         imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
         imputer.fit(X[:, 0:9])
         X[:, 0:9] = imputer.transform(X[:, 0:9])

In [33]: from sklearn.compose import ColumnTransformer
         from sklearn.preprocessing import OneHotEncoder
         ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [9])], remainder='passthrough')
         X = np.array(ct.fit_transform(X))

In [34]: print(X[4])

         [0.0 0.0 1.0 0.0 0.0 13.0 40.0 5.0 10.0 0.0 0.0 2.0 4.0 9.0 ' Female']

In [35]: from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         X[:,-1] = le.fit_transform(X[:,-1])

In [36]: print(X[4])

         [0.0 0.0 1.0 0.0 0.0 13.0 40.0 5.0 10.0 0.0 0.0 2.0 4.0 9.0 0]
```

- **DEALING WITH UNBALANCED DATASET**

  As the data was highly unbalanced, I used SMOTE (Synthetic Minority Over-sampling Technique) from sklearn to oversample the dataset with minority class. After this, I scaled the data using standard scalar except the sex column, so that no attribute overpowers other attributes.

- **DEALING WITH FEATURES**

  I tried removing certain attributes like F5, F6, F10 & F11 but it did not help in improving the accuracy or was decreasing the accuracy certain times.

- **CLASSIFICATION**

For classification I tried three types of classification which were Decision Tree, Support Vector Machine and XG Boost using libraries.

Decision Tree

For decision tree I set criterion as ‘entropy’, for information gain during split. The decision tree gave an F1 score of 0.85

Support Vector Machine

Applying SVM model with rbf kernel gave me an F1 score of 0.81

XGBoost

Next, I tried XGBoost for implementation of gradient boosted decision trees for better performance and speed. XGBoost gave an F1 score of 0.87

```
In [48]: from sklearn.metrics import f1_score
         f1_score(y_test, y_pred_dt, average='micro')

Out[48]: 0.857470334412082

In [49]: f1_score(y_test, y_pred, average='micro')

Out[49]: 0.8787756202804746

In [63]: f1_score(y_test, y_pred_svm, average='micro')

Out[63]: 0.819039913700108

In [51]: from sklearn.metrics import confusion_matrix, accuracy_score
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
         accuracy_score(y_test, y_pred)

         [[6378 1076]
          [ 722 6656]]

Out[51]: 0.8787756202804746
```

Based on these results, I selected XGBoost as the best model for prediction of test data.

The XGBoost really worked better than other models because it was better at handling the unbalanced data, XGBoost performs regularization to avoid overfitting, combined with parallel tree building which might have helped in better performance as compared to decision tree or SVM.