Credit Risk Prediction

Name: **Anisha Yidala** [redacted] Rank and F1score: **0.60**

Approach:

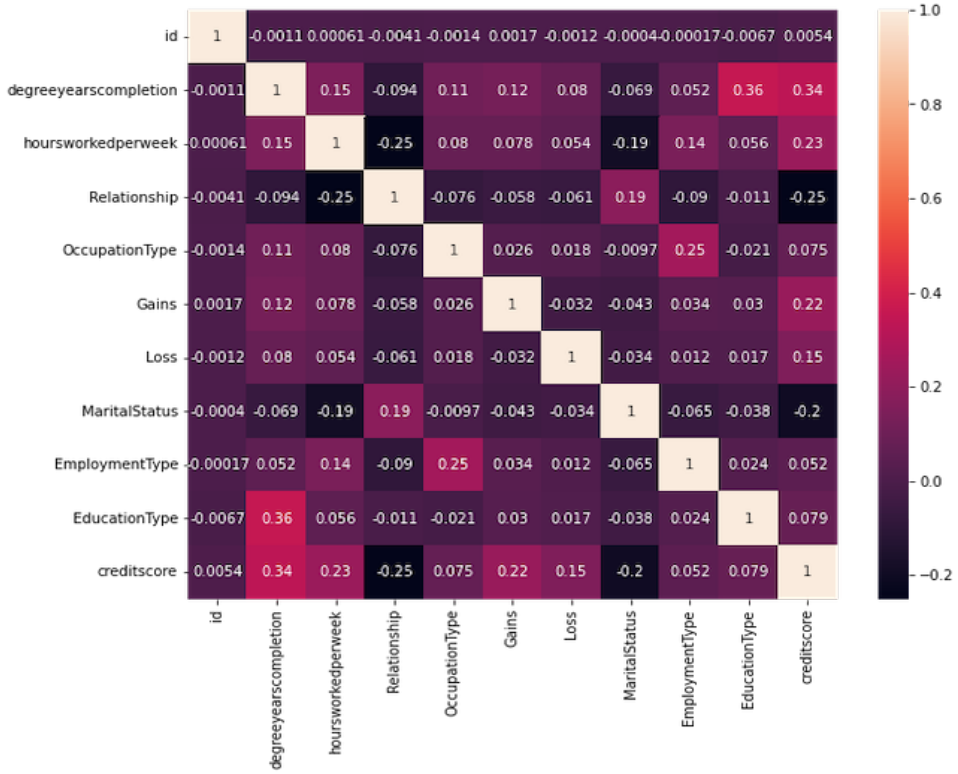
I have dealt with this project in two steps:

- Data Preprocessing
- Selecting a suitable model to best classify the given data

Data Preprocessing:

The given data is imbalanced, and it is also known that it has both continuous variables and categorical variables. These are the two main issues that must be taken care of in the preprocessing stage of this data.

- First step is to read the dataset and check for null values in the dataset. I have used the pandas library to read the csv file and then stored it in a data frame. To check for null values in the dataset I have used isnull( ).sum( ) function and concluded that there are no null values in both training data and test data.
- I have renamed the columns in both train data and test data according to the description given in the question for better/easy understanding of the data.
- I have used the correlation function corr( ) to compute the correlation among all the features in the training data and represented them in the form of a heatmap using the seaborn heatmap. The heatmap generated looks as shown below:



- I have eliminated columns id, Education Type F9, Marital Status F7 are three features that I have eliminated. I have eliminated id as it is unique for every entry of data and it is not very helpful in training any model to draw any relation between credit score and id. I have removed F9 education type as I have noticed from the correlation heatmap between the features generated earlier that F1 degree years completion and F9 Education type has the highest correlation among all the features. Removing the Marital Status has improved the results than when it was included on different models.
- Categorizing the continuous variables I have scaled the continuous columns in such a way that they all fall in the interval 0 -1 using the normalized( ) function. After normalizing the columns I divided the columns into 4 intervals using the pandas cut( ) function , which cut the continuous variables into bins of range (-0.1,0.25] , (0.25,0.50] , (0.50,0.75] , (0.75,1.0].
- I have used pandas.get_dummies( ) to eliminate dummies in the dataset.

Selecting a suitable model to best classify the given data:

I have tried three different models Decision Tree, Random Forest Classifier and XGB classifier. For me on trying various combinations of features for trying to train these three models I have seen that all three models decision tree, random forest and XGB classifier are classifying the test data up to 60% accurately. When I tried to do a test train split on the train data to do a cross validation and see which model is performing best for the combination of features that I found out are the best to train these models. I am using F1 score as a metric to measure the performance of these models. F1 score is the weighted average of precision and recall. I have split the train data in the ratio 70% training data and 30% test data using the test train split from sklearn model selection.

I have trained all three models on 70% of training data and predicted on the remaining 30% of the training data that we are using as test data. After predicting on the 30% of the data I have used this output to calculate the F1 score for each model. I have used f1score from sklearn metrics. The F1 score values for all the three models on the final combination of features I have used are as below: (implemented Cross validation.py file)

classifier	F1 score
Decision Tree	0.607
Random Forest	0.613
XGB Classifier	0.613

I have noticed that all three models are performing approximately same on this data. I have tried to use SVM model but noticed that its performance (F1 score dropped to 0.520) is much lesser than these and hence I have chosen Random Forest Classifier and the XGB classifier as those that are best classifying the given test data.
I have used random forest classifier and XGB classifier on the actual test data (implemented in main.py) and stored the output in the required format for miner submission when I submitted the files for both the files score is 0.60.

Random Forest classifier:
I have imported the Random Forest Classifier from sklearn ensemble. The random forest is known as an ensemble method because it combines the predictions of several models i.e. several decision trees by averaging the predictions of decision trees.
I have used a parameter n estimator the n estimators in random forest classifier represents the number of decision trees aggregate that we are considering to ensemble to train the model and predict the results. The n estimators value should be chosen carefully as it may lead to overfitting or underfitting of the model. After some trial and error for different values of n estimators I figured out 500 is giving the best F1 score without overfitting the model. As we increase the value of n estimators, I noticed that the F1 score is improving but a very high value of n estimators may lead the model to overfit.

XGB classifier:
I have imported the XGB classifier from xgboost. XGB is also an ensemble method like the random forest classifier. It has the **Gradient boosting** approach which is a method that goes through cycles to iteratively add models into an ensemble. Rather than training all the models in isolation of one another, each new model being is trained to correct the errors made by the previous ones, Models are added sequentially until no further improvement can be made. The gamma parameter can also help with controlling overfitting. It specifies the minimum reduction in the loss required to make a further partition on a leaf node of the tree. However, when I used this classifier on the given train data and performed cross validation by using the test train split I noticed that the F1 score is approximately same as for Random Forest classifier.

Measures Taken to deal with imbalance in the class distribution:
I have tried by changing the performance metric and experimenting algorithms that work best on imbalanced data . I have used decision tree and random forest models that learn from tree as they work well by learning a hierarchy of questions whish forces both classes to be addressed. I have also changed the performance metric as suggested from Accuracy to F score which is a the weighted average of precision and recall which proved to be a better measure as it has given very similar scores when measuring on test data.