MACHINE LEARNING ENGINEER NANODEGREE

CAPSTONE PROPOSAL

Title :- Prediction Of Consumer Credit Risk

- **Domain Background :-**

The domain background of this project is based on **credit risk** which can be defined as the probable risk of loss resulting from a borrower's failure to repay a loan or meet contractual obligations.Traditionally, it refers to the risk that a lender may not receive the owed principal and interest, which results in an interruption of cash flows and increased costs for collection.

In 2007, the U.S economy entered a mortgage crisis that caused panic and financial turmoil around the world. The financial markets became especially volatile, and the effects lasted for several years. The subprime mortgage crisis was a result of too much borrowing and flawed financial modeling, largely based on the assumption that home prices only go up. Greed and fraud also played important parts.

The goal of this project is to build an efficient tool for lending managers so that they can easily access the default risk of their clients.

Dataset :- https://www.kaggle.com/c/GiveMeSomeCredit/data

Related research work include:-

https://www.sciencedirect.com/science/article/pii/S095741741101342X?via%3Dihub [1]

http://cs229.stanford.edu/proj2014/Marie-Laure%20Charpignon,%20Enguerrand%20Horel,%20Flora%20Tixier,%20Prediction%20of%20consumer%20credit%20risk.pdf [2]

- **Problem Statement :-**

Predict the probability that somebody will experience a financial distress in the next two years using historical data of 1,50,000 borrowers that include features like age, debt ratio, monthly income, number of dependents, etc.

It is a binary classification problem and the output will be a binary value (either 0 or 1) ie

0 indicates that the individual **has not** experienced 90 days past due delinquency or worse, and

1 indicates that the individual **has** experienced 90 days past due delinquency or worse.

- **Datasets and Inputs :-**

The dataset for this project is obtained from a Kaggle competition **"Give Me Some Credit"**, the link of which is given below :-

https://www.kaggle.com/c/GiveMeSomeCredit/data

The provided dataset has 1,50,000 instances and 11 attributes. All the 11 attributes are described below :-

TARGET (OR DEPENDENT) VARIABLE

Label	Description
SeriousDlqin2yrs	Person experienced 90 days past due delinquency or worse.

FEATURES (OR INDEPENDENT) VARIABLES

Label	Description
RevolvingUtilizationOfUnsecured Lines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits.
Age	Age of borrower in years.
NumberOfTime30-59DaysPastDue NotWorse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years.
DebtRatio	Monthly debt payments, alimony,living costs divided by monthly gross income.
MonthlyIncome	Monthly Income
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards).
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due.

NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit.
NumberOfTime60-89DaysPastDue NotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years.
NumberOfDependents	Number of dependents in family excluding themselves. (spouse, children etc.)

In [5]:
```
# Initialising a dictionery to store feature_name as key and its corresponding outlier points as values.
d = {}

# Function to detect outliers.
def detect_outliers(feature):
    # For each feature find the data points with extreme high or low values
    #for feature in data.keys():

    # Calculate Q1 (25th percentile of the data) for the given feature
    Q1 = np.percentile(data[feature], 25)

    # Calculate Q3 (75th percentile of the data) for the given feature
    Q3 = np.percentile(data[feature], 75)

    # Use the interquartile range to calculate an outlier step (2 times the interquartile range)
    step = (Q3 - Q1) * 2

    outlier_points = data[data[feature] < Q1 - step]
    outlier_points = outlier_points.append(data[data[feature] > Q3 + step])
    d[feature] = outlier_points[feature].tolist()

    # Display the outliers
    print "Data points considered outliers for the feature '{}':- {}".format(feature, len(outlier_points))

# Calling the function
for i in data.keys():
    detect_outliers(i)
```

```
Data points considered outliers for the feature 'SeriousDlqin2yrs':- 10026
Data points considered outliers for the feature 'RevolvingUtilizationOfUnsecuredLines':- 528
Data points considered outliers for the feature 'age':- 2
Data points considered outliers for the feature 'NumberOfTime30-59DaysPastDueNotWorse':- 23982
Data points considered outliers for the feature 'DebtRatio':- 30815
Data points considered outliers for the feature 'MonthlyIncome':- 0
Data points considered outliers for the feature 'NumberOfOpenCreditLinesAndLoans':- 1898
Data points considered outliers for the feature 'NumberOfTimes90DaysLate':- 8338
Data points considered outliers for the feature 'NumberRealEstateLoansOrLines':- 473
Data points considered outliers for the feature 'NumberOfTime60-89DaysPastDueNotWorse':- 7604
Data points considered outliers for the feature 'NumberOfDependents':- 0
```

All the features have numerical values (both int and float) and except for “Age” no feature seems to be normalized. After analyzing the results from collections.Counter() and Tukey’s method of detecting outliers, I concluded that all of the features contain outliers.

Two of the features namely “MonthlyIncome” and “NumberOfDependents” also contains null values. I also analyzed the target variable and noticed that it is heavily biased as out of 1,50,000 instances, only 10026 (6.684%) instances has value equal to 0.

Also Kaggle does not provide testing data with output labels, so I have to split the training data into training and testing subsets. For this I will use sklearn.model_selection.train_test_split() to split the data into training and testing subsets. I will use a 80%-20% split for training and testing ie 80% of the data will be used for training purpose and the rest 20% for testing.

- **Solution Statement** :-

This is a binary classification problem and hence the most suitable algorithms for the project could be :-

1. **Logistic Regression :-** It is a very simple classification algorithm which is used to predict a binary outcome(0/1, no/yes, false/true). Cost function for logistic regression can be written as :-

   $J(\theta) = -(1/m) \sum[y(i) * \log(h\theta(x(i))) + (1-y(i)) * \log(1-h\theta(x(i)))]$

2. **Gradient Boosting Classifier :-** Gradient Boosting trees are very robust to outliers and they improve the accuracy of a function through incremental minimization of cost function.
3. **Random Forests :-** Random forest creates many decision trees using different subsets of training data and averages them to obtain a more accurate or strong model.
4. **XGBoost Classifier :-** It is just an implementation of gradient boosting trees and is mainly designed for speed and performance. It further helps to reduce overfitting or variance in the data. The main purpose of using xgboost is that it is very fast and efficient.
5. **Multi-Layered Perceptron :-** It is basically a feedforward artificial neural network which consists of at least three layers of nodes and except for the input nodes, each node is a neuron that uses a non-linear activation function. Using MLP I will try to find non-linearity in the data.

- **Benchmark Model :-**

For defining a benchmark model, I will use Logistic Regression on original dataset without any pre-processing or feature engineering. Although before defining the benchmark, I have to process null values in the data because the model cannot be trained on null values. I cannot remove all the rows that contain null value in any column as it might make the target variable more biased. Therefore, I will use a very simple method and replace all the null values with median of the data because median is less sensitive to outliers as compared to mean.

- **<u>Evaluation metric</u> :-**

The evaluation metric that will be used in this problem is:-

1. **AUC :-** It stands stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve. AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example

- **<u>Project Design</u> :-**

I will use the following steps to derive a solution to the problem :-

1. **Data Exploration :-** Here, I will try to describe and analyze the data statistically. I will try to find the number of outliers ,degree of skewness and null values in the data.
2. **Data Visualization :-** I will try to visualize features and target variable using pie charts, bar graphs, histogram, etc and try to find correlated features and visible pattern in the data.
3. **Data Preprocessing :-** I will separate the target variable and independent variables and split the data in training and testing subsets.Each feature will be processed and outliers and null values will be removed.
4. **Feature Engineering :-** Here, I will try to use PCA to explain variability in the data and to reduce dimension and thus try to improve the overall score.
5. **Model selection :-** Various models will be selected and experimented to find out the best model.
6. **Hyper parameter tuning :-** The best model selected will be tuned on various other parameters to further improve the performance.
7. **Testing and Evaluation :-** The tuned model will be tested on testing data and the result will be evaluated.

- **References :-**

1. https://www.sciencedirect.com/science/article/pii/S095741741101342X?via%3Dihub
2. http://cs229.stanford.edu/proj2014/Marie-Laure%20Charpignon,%20Enguerrand%20Horel,%20Flora%20Tixier,%20Prediction%20of%20consumer%20credit%20risk.pdf
3. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/
4. https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
5. https://xgboost.readthedocs.io/en/latest/python/python_api.html
6. http://datapigtechnologies.com/blog/index.php/highlighting-outliers-in-your-data-with-the-tukey-method/
7. https://www.investopedia.com/terms/c/creditrisk.asp