CS 418 Final Project Report

Problem Selection

Our problem was predicting the probability that someone will experience financial distress in the next two years. We assessed the risk factors associated with low credit scores and found patterns that would help us predict financial distress. Being able to predict financial distress would be useful to improving the already-existing credit score algorithm banks use to decide whether a loan should be granted.

In this project we have built a random forest classifier to compute the truth value of financial distress that a customer may face in the next 2 years and predict the corresponding probability using different attributes of customers. We then built a clustering model to cluster customers to better understand the relation between their attributes and financial distress.

Data Collection

The main datasets we used were the cs-test.csv and cs-training.csv files from: https://www.kaggle.com/c/GiveMeSomeCredit/data. Our datasets included a fairly large amount of variables describing each participant's current financial situation. Variables include demographic information such as age and number of dependents, as well as more finance-specific information like 'Number of Times 30-59 Days Past Due' and 'Number of Open Credit Lines and Loans.' One issue we had with the raw data was that some of it was skewed.

Data Preparation

Handling Missing Data

On preliminary exploration, we discovered that our dataset had some missing values in the variables, *'MonthlyIncome*' and *'NumberOfDependents'*

SeriousDlqin2yrs	0.000000
RevolvingUtilizationOfUnsecuredLines	0.000000
age	0.000000
NumberOfTime30-59DaysPastDueNotWorse	0.000000
DebtRatio	0.000000
MonthlyIncome	0.198207
NumberOfOpenCreditLinesAndLoans	0.000000
NumberOfTimes90DaysLate	0.000000
NumberRealEstateLoansOrLines	0.000000
NumberOfTime60-89DaysPastDueNotWorse	0.000000
NumberOfDependents	0.026160

The percentage of missing data being relatively small compared to the size of the dataset, we decided to remove the observations with missing values.

Removing NaNs and cleaning data

```
data.dropna(axis=0,how='any',subset=['NumberOfDependents'],inplace=True)
data.dropna(axis=0,how='any',subset=['MonthlyIncome'],inplace=True)
data.reset_index()
```

Creating new features

Some variables in the dataset were not useful for training in their current state. To work around this issue, we performed Feature Extraction to create more useful features. The new features can be seen in the snippet of code below.

Creating new features and cleaning more

```
train_frame['MonthlyDebt'] = train_frame['DebtRatio']*train_frame['MonthlyIncome']
train_frame['NumOfPastDue'] = train_frame['NumberOfTimes90DaysLate']+train_frame['NumberOfTime60-89DaysPastDueNotWorse'] +train_frame['NumberOfTime30-59DaysPastDueNotWorse']
train_frame['MonthlyBalance'] = train_frame['MonthlyIncome']-train_frame['MonthlyDebt']
train_frame['NumOfOpenCreditLines'] = train_frame['NumberOfOpenCreditLinesAndLoans']-train_frame['NumberRealEstateLoansOrLines']
train_frame['IncomePerPerson'] = train_frame['MonthlyIncome']/(train_frame['NumberOfDependents']+1)

# We need only the observations where MonthlyBalance is positive
train_frame['MonthlyBalance'][train_frame['MonthlyBalance'] <= 0] = 1
```

Outliers detection and Removal

	count	mean	std	min	25%	50%	75%	max
SeriousDlqin2yrs	120269.00000	0.06949	0.25428	0.00000	0.00000	0.00000	0.00000	1.00000
RevolvingUtilizationOfUnsecuredLines	120269.00000	5.89987	257.04068	0.00000	0.03508	0.17728	0.57943	50708.00000
age	120269.00000	51.28979	14.42668	0.00000	40.00000	51.00000	61.00000	103.00000
NumberOfTime30-59DaysPastDueNotWorse	120269.00000	0.38177	3.49923	0.00000	0.00000	0.00000	0.00000	98.00000
DebtRatio	120269.00000	26.59878	424.44646	0.00000	0.14339	0.29602	0.48256	61106.50000
MonthlyIncome	120269.00000	6670.22124	14384.67422	0.00000	3400.00000	5400.00000	8249.00000	3008750.00000
NumberOfOpenCreditLinesAndLoans	120269.00000	8.75847	5.17284	0.00000	5.00000	8.00000	11.00000	58.00000
NumberOfTimes90DaysLate	120269.00000	0.21192	3.46528	0.00000	0.00000	0.00000	0.00000	98.00000
NumberRealEstateLoansOrLines	120269.00000	1.05452	1.14927	0.00000	0.00000	1.00000	2.00000	54.00000
NumberOfTime60-89DaysPastDueNotWorse	120269.00000	0.18783	3.44790	0.00000	0.00000	0.00000	0.00000	98.00000
NumberOfDependents	120269.00000	0.85183	1.14839	0.00000	0.00000	0.00000	2.00000	20.00000
MonthlyDebt	120269.00000	2117.74900	4196.59602	0.00000	595.28279	1619.79100	2879.73087	478450.55916
NumOfPastDue	120269.00000	0.78152	10.34807	0.00000	0.00000	0.00000	0.00000	294.00000
MonthlyBalance	120269.00000	4688.58980	14104.90092	0.50000	1941.25336	3567.02273	5867.34808	3004327.00211
NumOfOpenCreditLines	120269.00000	7.70396	4.79662	0.00000	4.00000	7.00000	10.00000	57.00000
IncomePerPerson	120269.00000	4509.23510	8884.01804	0.00000	1960.00000	3333.33333	5658.00000	1794060.00000

The data has outliers that are meaningless or unreasonable with respect to the problem. These outliers were further analysed.

```
print('Percentage of outliers in age: {}'.format(train_frame.query('age == 0 or age >= 99').size/len(train_frame)))
print('Percentage of outliers in RevolvingUtilizationOfUnsecuredLines: {}'.format(train_frame.query('RevolvingUtilizationOfUnsecuredLines > 5').size/len(train_frame)))
print('Percentage of outliers in DebtRatio: {}'.format(train_frame.query('DebtRatio > 9000').size/len(train_frame)))
```

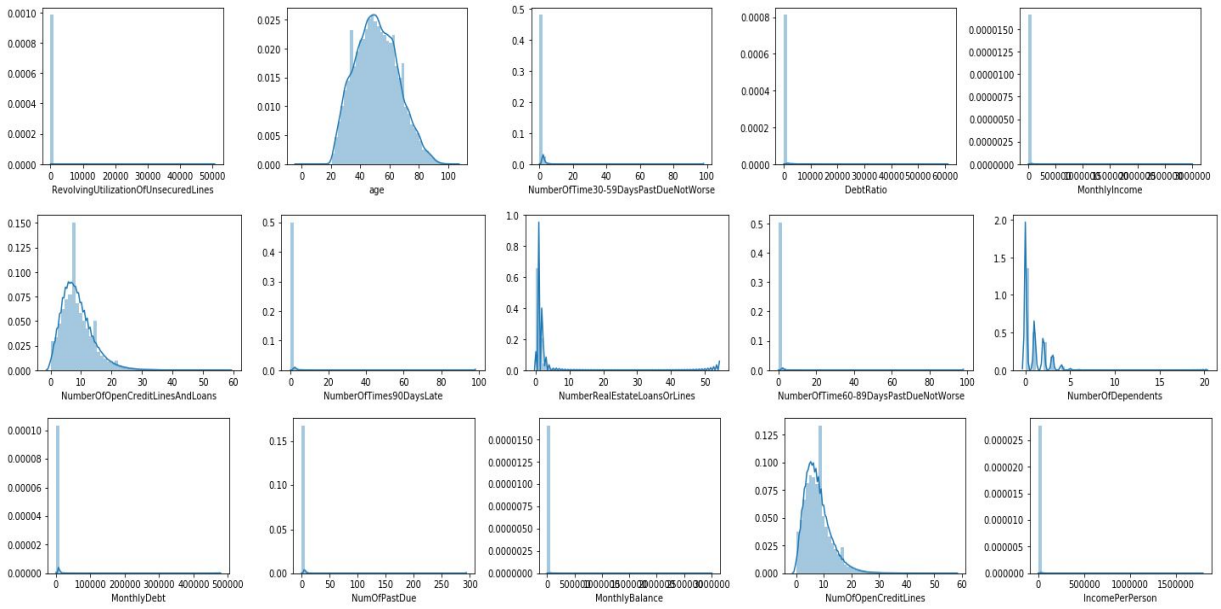
```
Percentage of outliers in age: 0.001330351129551256
Percentage of outliers in RevolvingUtilizationOfUnsecuredLines: 0.024212390557832857
Percentage of outliers in DebtRatio: 0.0029267724850127633
```

The percentages of outliers in the above columns do not seem to be large. Thus we removed these observations using a query below.

train_frame = train_frame.query('age > 0 and age < 99').query('RevolvingUtilizationOfUnsecuredLines < 5').query('DebtRatio < 9000')

Handling skewed data

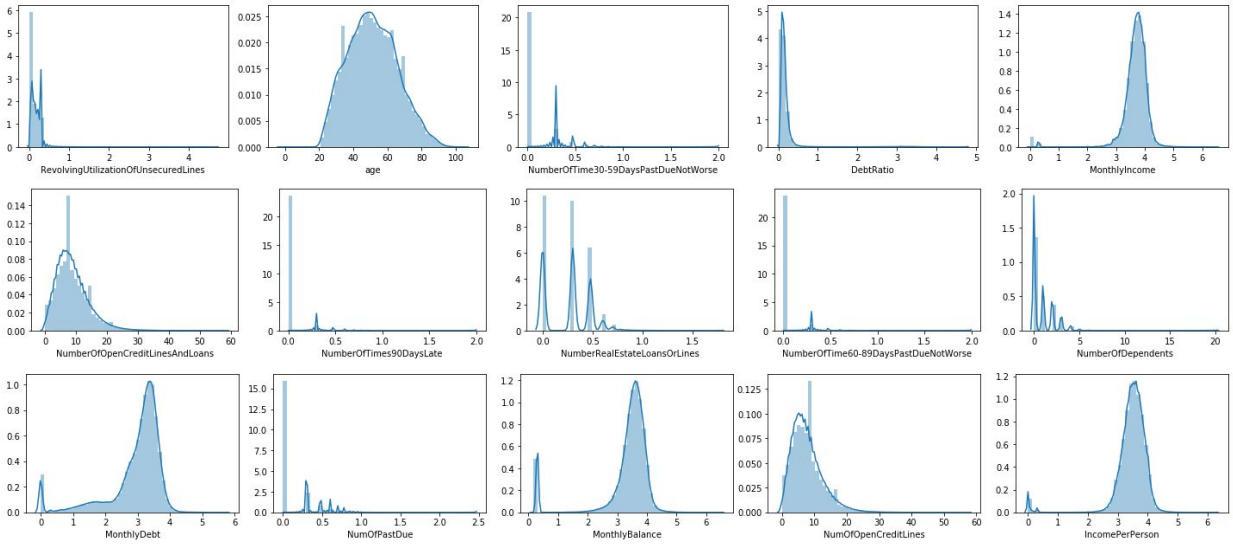
We plotted the distribution of each variable in the data and noticed some of our data was skewed. Below are the plots that give an intuition of how skewed each variable is.



To make the distribution of some variables less skewed, we used log transform as shown below

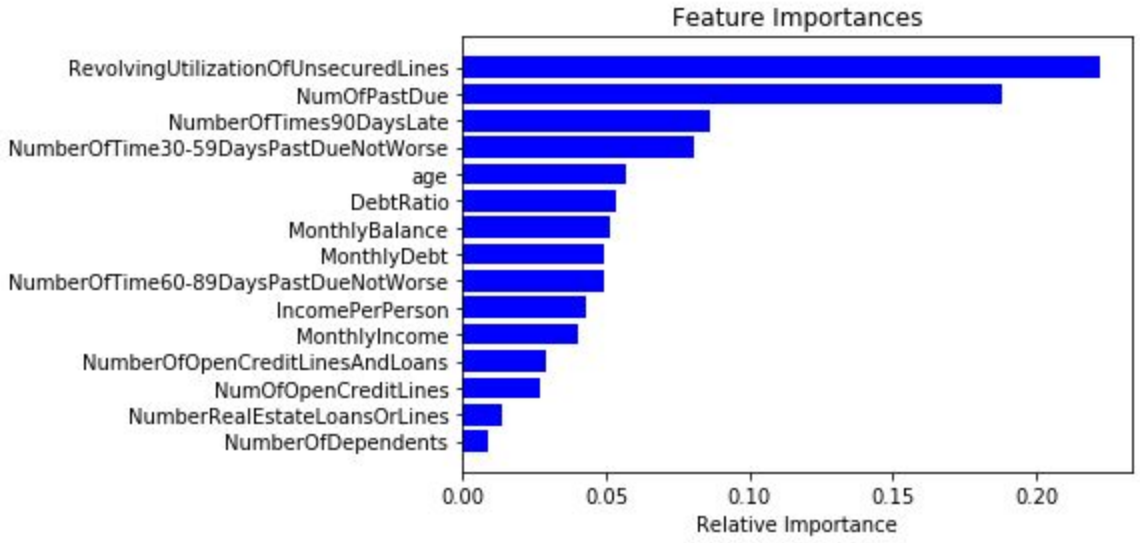
```
for column in ['RevolvingUtilizationOfUnsecuredLines', 'NumberOfTime30-59DaysPastDueNotWorse', 'NumberOfTimes90DaysLate',
               'NumberOfTime60-89DaysPastDueNotWorse', 'NumOfPastDue', 'MonthlyDebt','MonthlyIncome', 'DebtRatio',
               'NumberRealEstateLoansOrLines', 'IncomePerPerson', 'MonthlyBalance']:
    train_frame[column] = numpy.log10(1 + train_frame[column].values)
```

After the transformation, the values of each variable became more evenly distributed than before.



Data Exploration

To better understand each variable's relevance to financial distress, we created a bar graph to visualize the relative importance of each variable. The bar graph was based on data from a Random Forest Classifier trained on all the predictor variables. Visualizing the data this way helped us understand which variables greatly influence the participants' financial situation, and which were less influential.



Data Modeling

Classification

We used classification to classify each observation as whether individual pertaining to that observation will face a serious financial distress in the next 2 years. We have also predicted probabilities of such events occuring. We created an 'out.csv' file that displays each entry's truth value for financial distress and the corresponding probability.

Splitting Dataset

We split the data using the hold-out method to train and evaluate the model. We used 80% of the data for training and 20% for evaluation.

```
def split_dataset(features):
    x = train_frame[features]
    y = train_frame['SeriousDlqin2yrs']
    return train_test_split(x, y, test_size=0.2, random_state=0)
```

Hyper-parameter Tuning

To find the optimal hyper-parameters for our model, we used *GridSearchCV* to perform hyper-parameter estimation and tuning using cross-validation on the training dataset.

```
def tune_hyper_parameters(x_train, y_train):
    rf = RandomForestClassifier(oob_score=True, \
                                min_samples_split=2, \
                                min_samples_leaf=50, \
                                n_jobs=-1, \
                                #class_weight="balanced",\
                                class_weight="balanced_subsample", \
                                bootstrap=True\
                                )
    param_grid = {"n_estimators": [90, 100, 110, 130], "max_features": [2, 3, 4], "min_samples_leaf":[50]}
    grid_search = GridSearchCV(rf, cv=10, scoring='roc_auc', param_grid=param_grid, iid=False)
    grid_search.fit(x_train, y_train)
    pprint(grid_search.best_params_)
    pprint(grid_search.best_score_)
tune_hyper_parameters(x_train, y_train)
```

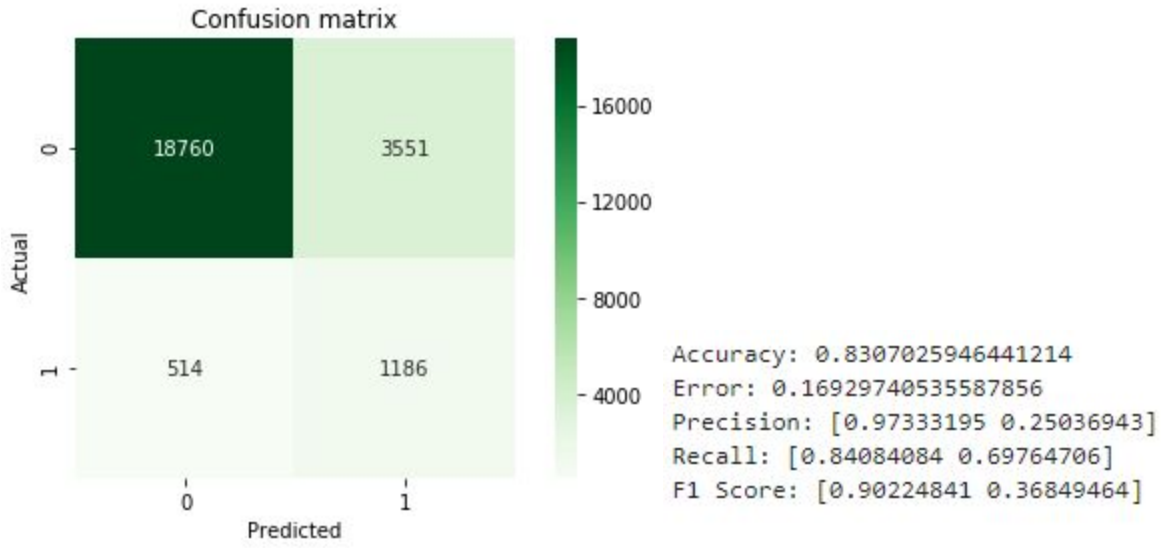
```
{'max_features': 2, 'min_samples_leaf': 50, 'n_estimators': 130}
0.8525137261197562
```

Model Training and Evaluation

We used the hyper-parameters obtained in the previous step for our model. The features used are the top 12 with respect to their relative importance.

```
features = ['RevolvingUtilizationOfUnsecuredLines','MonthlyDebt', 'MonthlyIncome', 'DebtRatio','age',
            'IncomePerPerson', 'MonthlyBalance', 'NumOfOpenCreditLines','NumberOfTime30-59DaysPastDueNotWorse',
            'NumberOfTime60-89DaysPastDueNotWorse', 'NumberOfTimes90DaysLate', 'NumOfPastDue']
x_train, x_test, y_train, y_test = split_dataset(features)
# x_train, y_train = cc_under(x_train, y_train)
x_train, x_test = scalar_transform(x_train, x_test)
rf = train(x_train, y_train)
```

For classification, we used a random forest classification model with optimal hyper-parameters and most important features. Below are the confusion matrix and evaluation metrics for the trained model. The evaluation was done with the test dataset obtained previously by splitting the whole dataset using the hold-out method.




```
Accuracy: 0.8307025946441214
Error: 0.16929740535587856
Precision: [0.97333195 0.25036943]
Recall: [0.84084084 0.69764706]
F1 Score: [0.90224841 0.36849464]
```

Predicting results on the test dataset

We have used this model to predict the class and probability of distress for each observation in the test dataset and saved the file as ‘out.csv’

	A	B	C
1		Probability	SeriousDlqin2yrs
2	1	0.292979656	0
3	2	0.2517307752	0
4	3	0.2104809031	0
5	4	0.3829768764	0
6	5	0.4777545519	0
7	6	0.2507128074	0
8	7	0.1511555229	0
9	8	0.3146055025	0
10	9	0.1131983272	0
11	10	0.9266285359	1

As shown in the table above, person 1 - 9 is classified to not have financial distress in the next 2 years. While person 10 will most likely have financial distress after the classification model.

Clustering

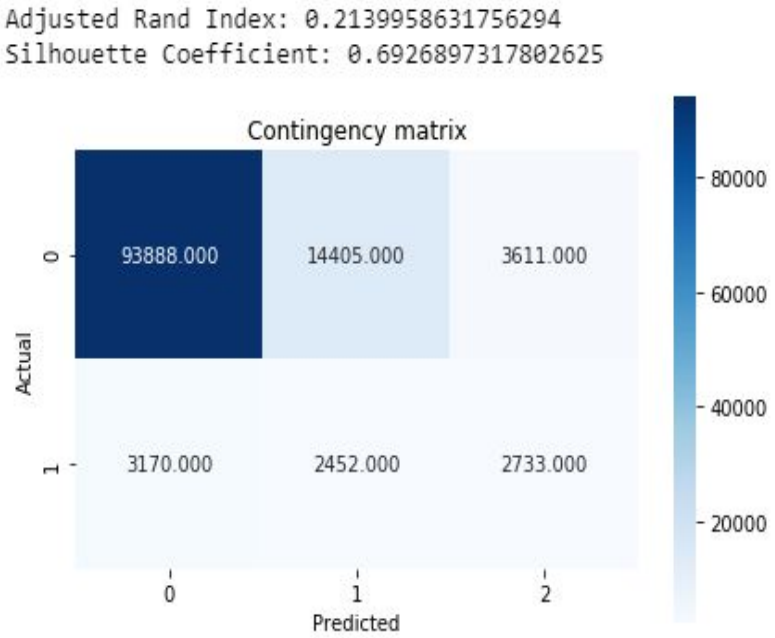
For this task we used the same cleaned and prepared data as we did for building the Classification model. However, we did not need as many features as we did for classification. So we only selected few of the top features from the results obtained previously.

Training and Evaluating Clustering Model

We used K-Means as the clustering technique to cluster the observations as *High risk, Moderate risk* and *Low risk*.

```
def k_means(X, y_test):
    clustering = KMeans(n_clusters = 3, init = "k-means++", n_init = 4, max_iter=800, random_state = 0, algorithm = "auto").fit(X)
    clusters = clustering.labels_
    cont_matrix = metrics.cluster.contingency_matrix(y_test, clusters)
    ax = sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
    bottom, top = ax.get_ylim()
    ax.set_ylim(bottom + 0.5, top - 0.5) #to fix glitch
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title('Contingency matrix')
    plt.tight_layout()
    print(evaluate(X, y_test, clusters))
    return clusters
```

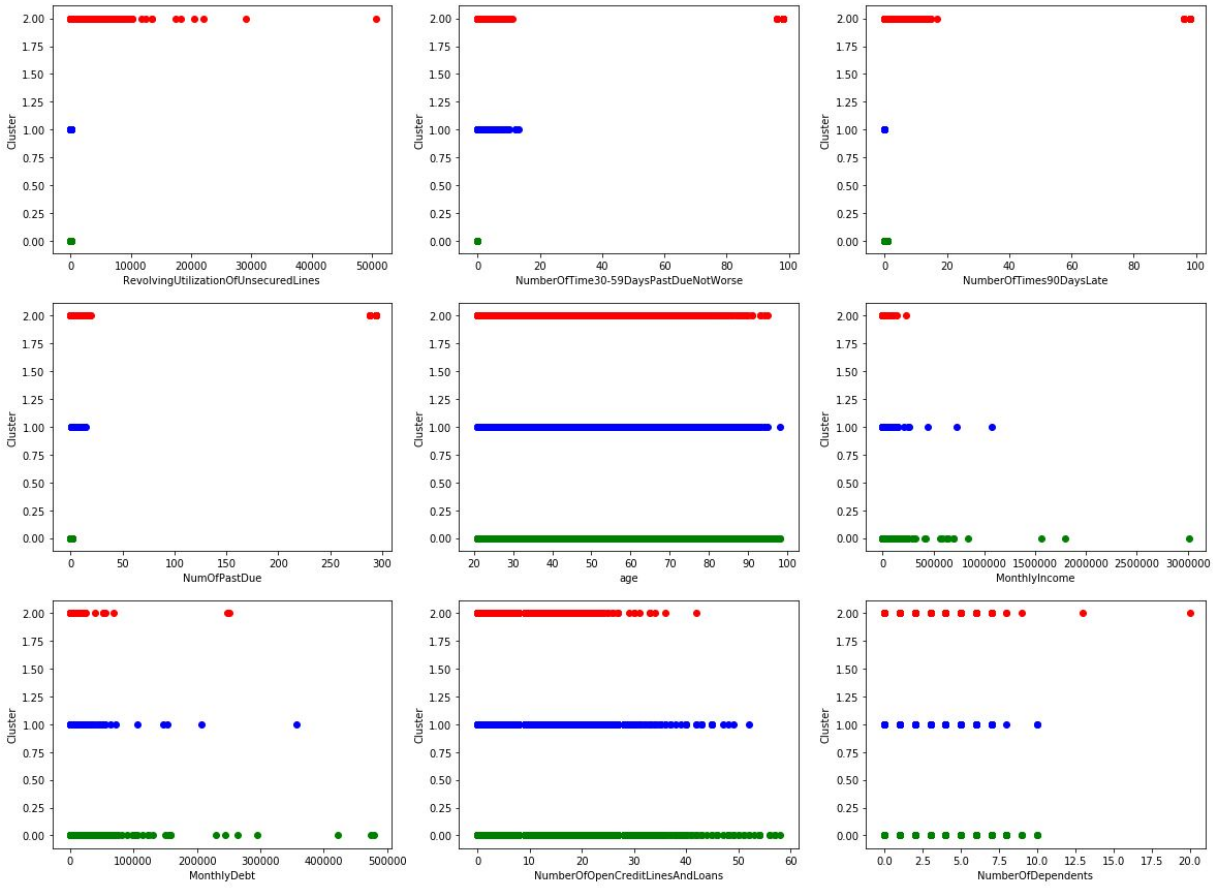
The contingency matrix and the evaluation metrics of the trained model are given below.



The model seems to have a moderately high Silhouette Coefficient, *i.e*, There is high cohesion among the observations in a single cluster and good separation between observations of different clusters.

Cluster Analysis

To further understand how different attributes of an individual influence the risk involved in sanctioning credit to them, we created scatter plots using the said attributes and corresponding cluster of each observation.



Cluster	Risk	Color
0	Low	Green
1	Medium	Blue
2	High	Red

Presentation of Results

Using the scatter plots above, we were able to draw some conclusions and interpretations on the effect of customer attributes on the probability of them facing financial distress in the next 2 years. We have created a table containing these conclusions.

Attribute	Definition	Observation from Each Cluster
RevolvingUtilizationOfUnsecuredLines	Percentage of credit utilized	Observations in the low risk and medium risk clusters seem to have a low value. However, observations in the high risk cluster have extremely high values for this attribute
NumberOfTime30-59DaysPastDueNotWorse	Number of payments late by 30 to 59 days	The values for this attribute seem to be increasing as the risk increases.
NumberOfTimes90DaysLate	Number of payments late by more than 90 days	As expected, the values for this attribute seem to be increasing as the risk increases. Credit risk associated with a person is high when they are always late on their payments
NumOfPastDue	Total number of previous late payments	People who make late payments consistently tend to have a higher credit risk
Age	Age of a person	Age is evenly distributed across all clusters. All clusters contain all age groups.
MonthlyIncome	Monthly income of a person	Monthly income weakly decreases as the risk increases. People with higher income have the ability to make payments on time
MonthlyDebt	Monthly debt payments	Contraty to expectations, observations with high risk have more debts. One interpretation is that people with high debts also have high income. People with high income tend to get debts sanctioned more easily.
NumberOfOpenCreditLinesAndLoans	Number of credit and loan lines	Like MonthlyDebt, people in the low risk region have more loans and credit lines. This can be interpreted the same way as MonthlyDebt. People with high income do tend to have many lines of credit.
NumberOfDependents	Number of depends of each individual	Credit risk weakly increases with respect to increase in number of dependents. This makes sense because they have too many people to take care of to make payments on time.