PROJECT REPORT

Group No: Project Group 1

Student Names: Anish Nair, Sreenihar Akshinthala, Vishwajith Subhash

ABSTRACT:

Financial firms often tend to scuffle around in this competitive world to gain customers, in order to be at the top of the game. Different strategies are put into use to achieve a firm's goals while another one just across the street use the same tactics. To stay ahead of their contenders, we propose an organization to utilize data available to make strategic decisions.

Using machine learning, a company can know beforehand if a customer is willing to make a deposit in their bank or not. This reduces the time and energy spent on trying to convince one person, and not lose out on potential clients.

INTRODUCTION:

The bank marketing dataset allows us to perform analysis and modelling, giving us insights into how frequently a telemarketing campaign needs to function to attain customers. The data present contains information about each potential client, such as their occupation, age, and the credit status. All this information is used to predict if the client will open up a account at the bank. The dataset also contains internal information such as consumer indexes and the monthly indicators for each client.

One of the biggest issues we faced during the course of the project was that the data had a pretty big class imbalance. Approximately a 90/10 split towards the False category in the output variable.

DATA DESCRIPTION:

The following variables contain client information collected by the bank in question.

Variable Name	Variable Type	Description
age	Numerical	Age of the client
job	Categorical	Type of job
marital	Categorical	Marital status of the client
education	Categorical	Education level of the client
default	Categorical	Does the client have credit in default?
housing	Categorical	Does the client have a housing loan?

loan	Categorical	Does the client have a personal loan?
contact	Categorical	Contact Communication type
month	Categorical	Last contact month of the year
day_of_week	Categorical	Last contact day of the week
duration	Numeric	Last contact duration in seconds

The following variables contain information dedicated to the current campaign and the previous campaign

Campaign	Numeric	Number of contacts performed during this campaign for this particular client
pdays	Numeric	Number of days that passed by after the client was last contacted from a previous campaign
previous	Numeric	Number of contacts performed before this campaign for this client
poutcome	Categorical	Outcome of the previous marketing campaign
y (Output Variable)	Categorical	Has the client subscribed to a term deposit?

METHODS:

We are aiming to do a binary classification problem, which will return an output either yes or no indicating whether the client will subscribe to a term deposit or not. For this task we have implemented the following classification techniques:

i) Logistic Regression
ii) SVM
iii) Feed-Forward Neural Network

Logistic Regression:

We have used logistic regression as our baseline model. We have implemented Logistic Regression in 2 ways:

1) Without oversampling – Attained accuracy 89%, and F1 Score 0.28
2) With SMOTE – Attained accuracy 76%, and F1 Score 0.38

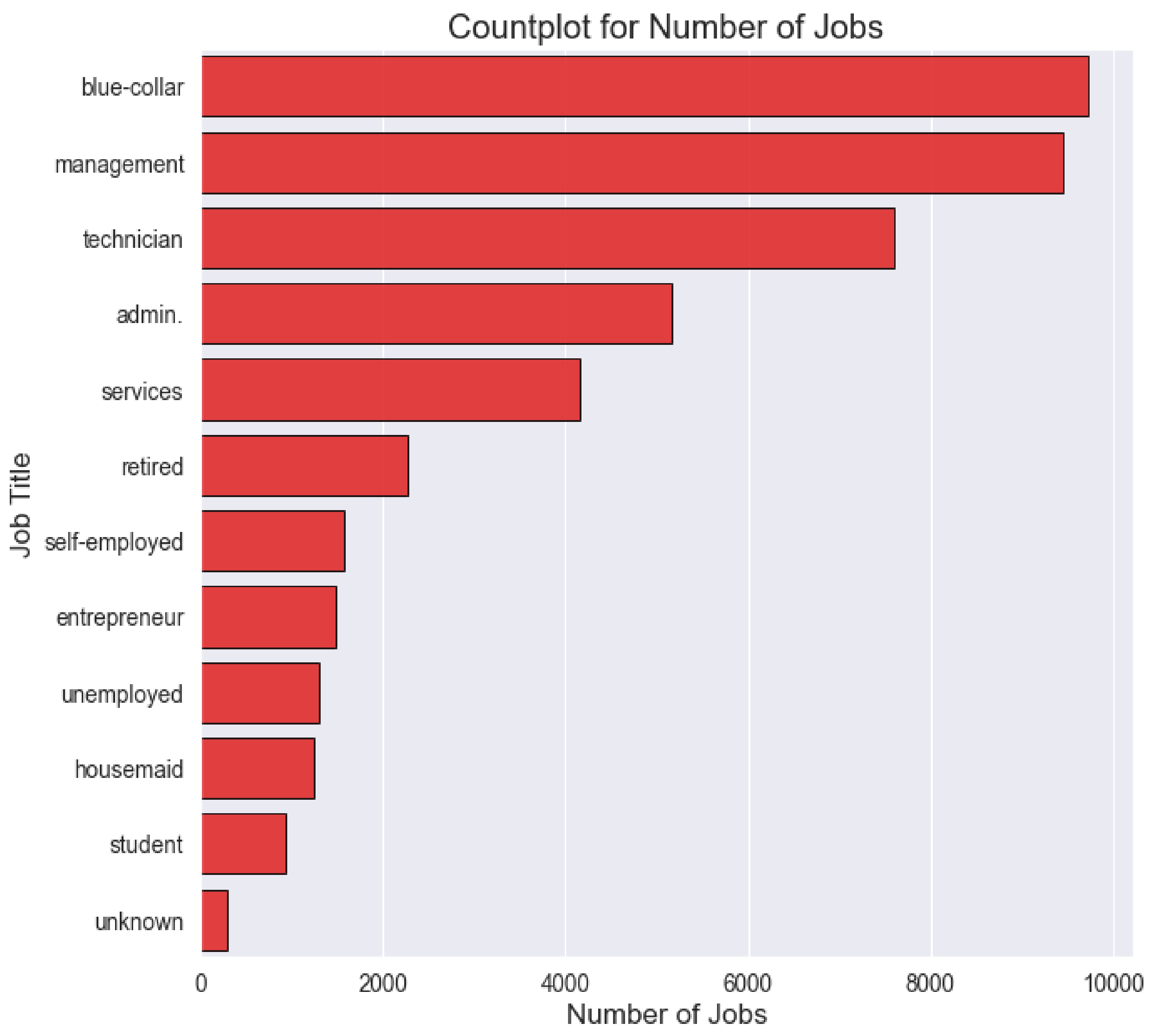
SVM:

SVM with RBF kernel allows us to classify model which is separated by a non-linear hyperplane. But, this model fails to achieve the optimum results as the accuracy and F1-score are 0.49 and 0.49 respectively.

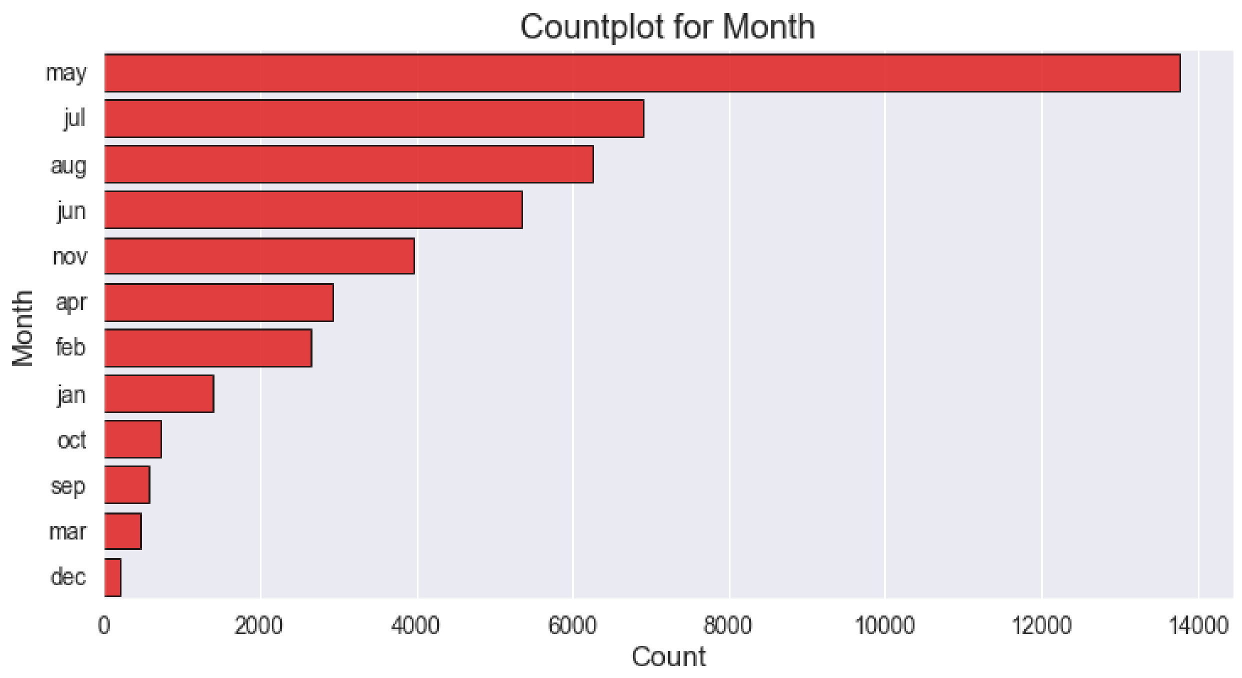
Neural Network:

We have implemented the feed-forward neural network with an accuracy of 0.89 and F1 score of 0.39. But if we use SMOTE, it is observed that the model is better optimized, as there is no overfitting, as we can notice in the Validation Error vs Training Error chart. The validation error remains constant, whereas the training error keeps reducing until it reaches it's optimum. The accuracy with SMOTE is 0.88 and the F1 score is 0.37.

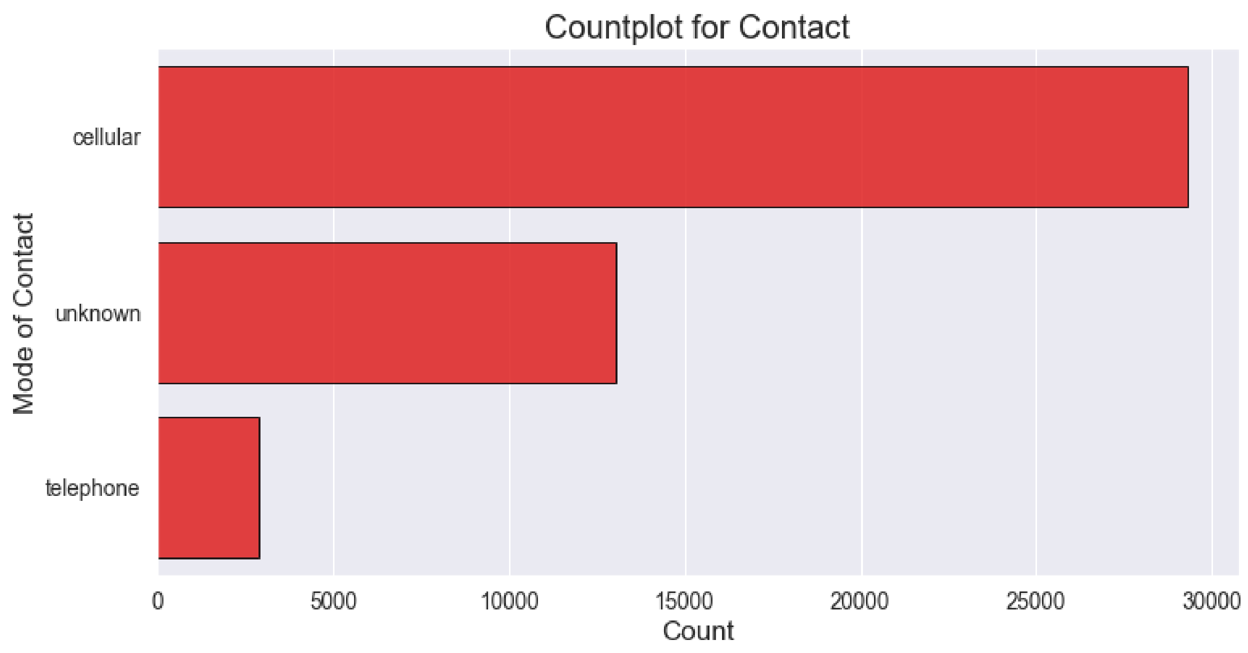
EDA



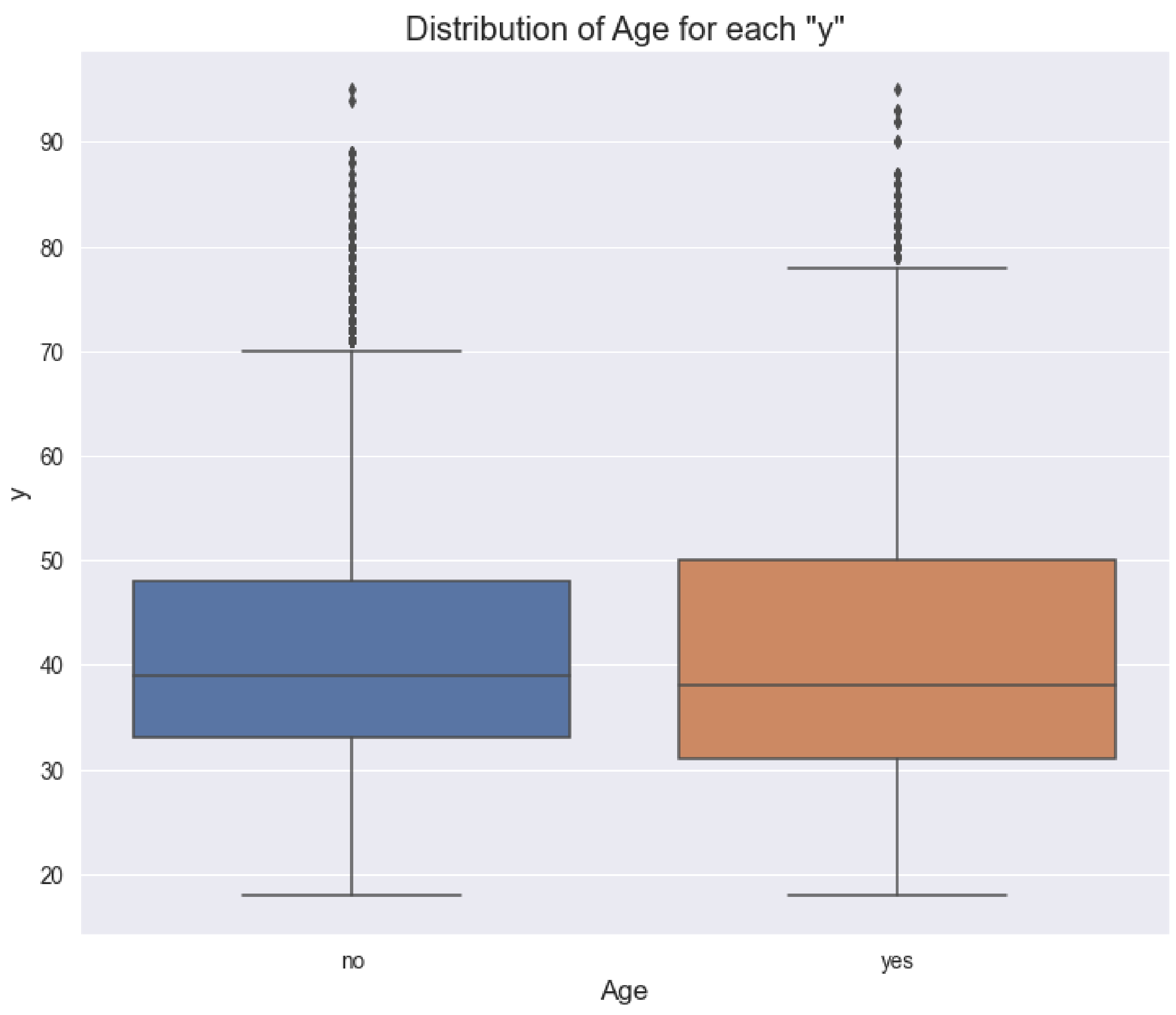
The above bar chart visualizes the count of the types of jobs that all the clients perform.



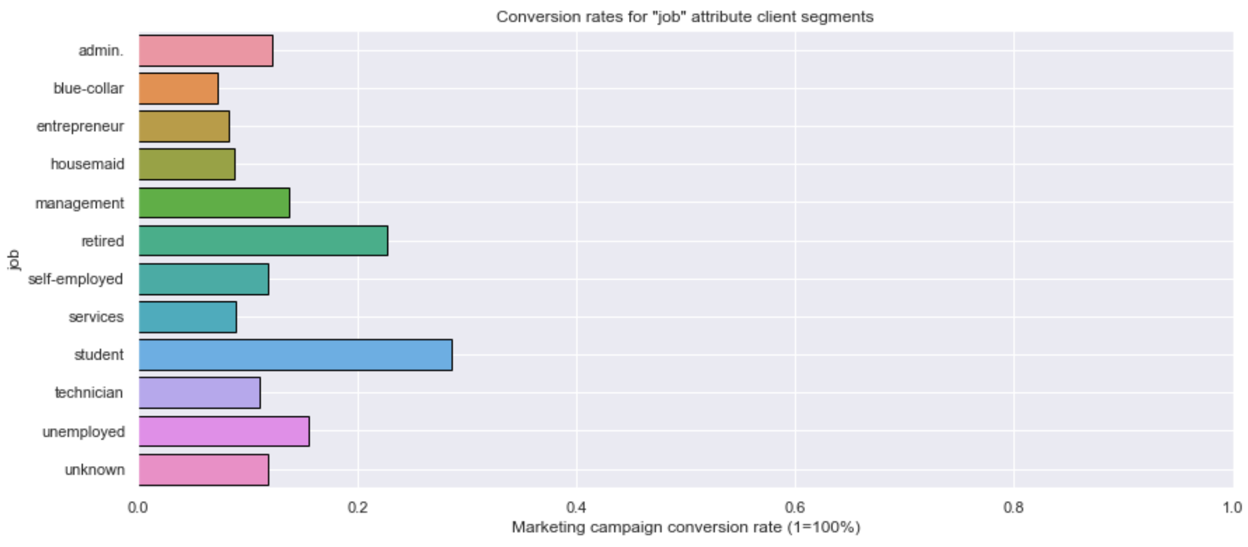
The distribution of the number of clients agreeing to opening a bank deposit in each month is depicted in the above bar chart. The month of May certainly seemed to be the most important month for the bank.



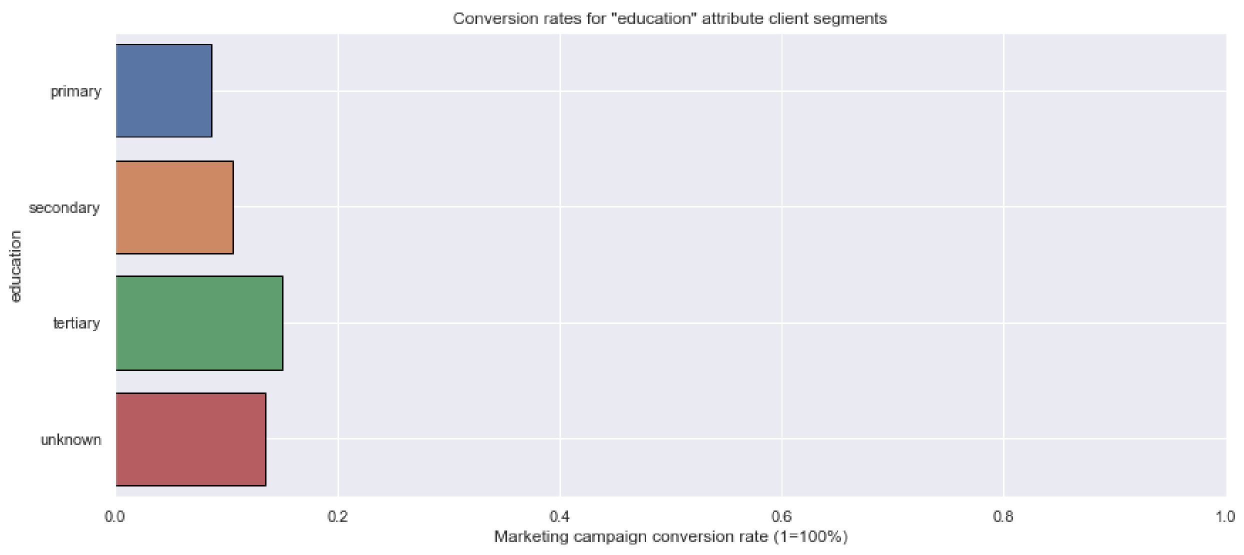
The method of communication through a cellular device seems to be the popular one for the bank.



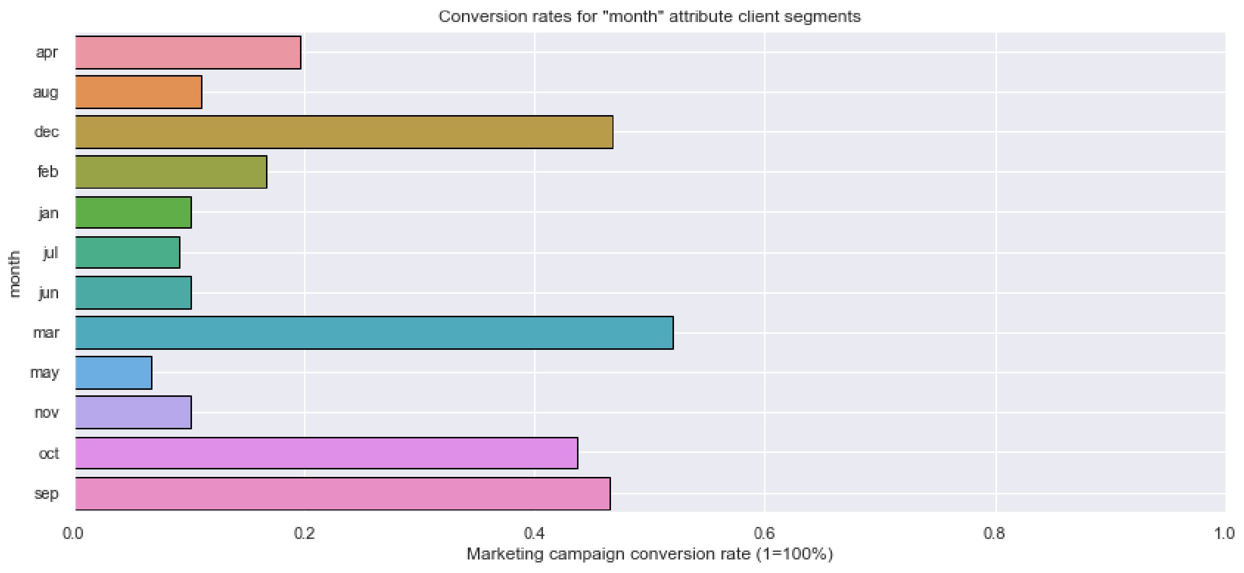
The minimum values and the means of the ages of the people for people who opened a deposit, and the ones who didn't is approximately the same. The maximum values differ by around 8 years.



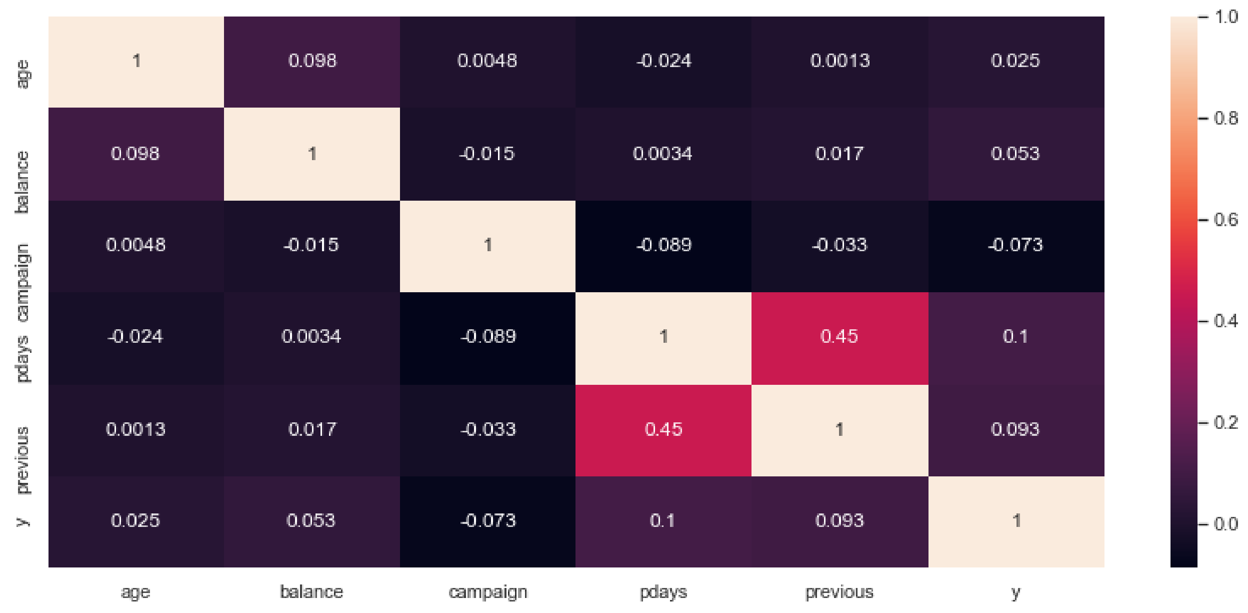
The conversion rates of campaigns leading to people creating deposit accounts based on their job title is depicted above. The blue-collar job-holders seem to be the least interested in opening an account.



Contrary to the general thought, clients holding primary education seem to be staying away from the need to open a deposit account at a bank.

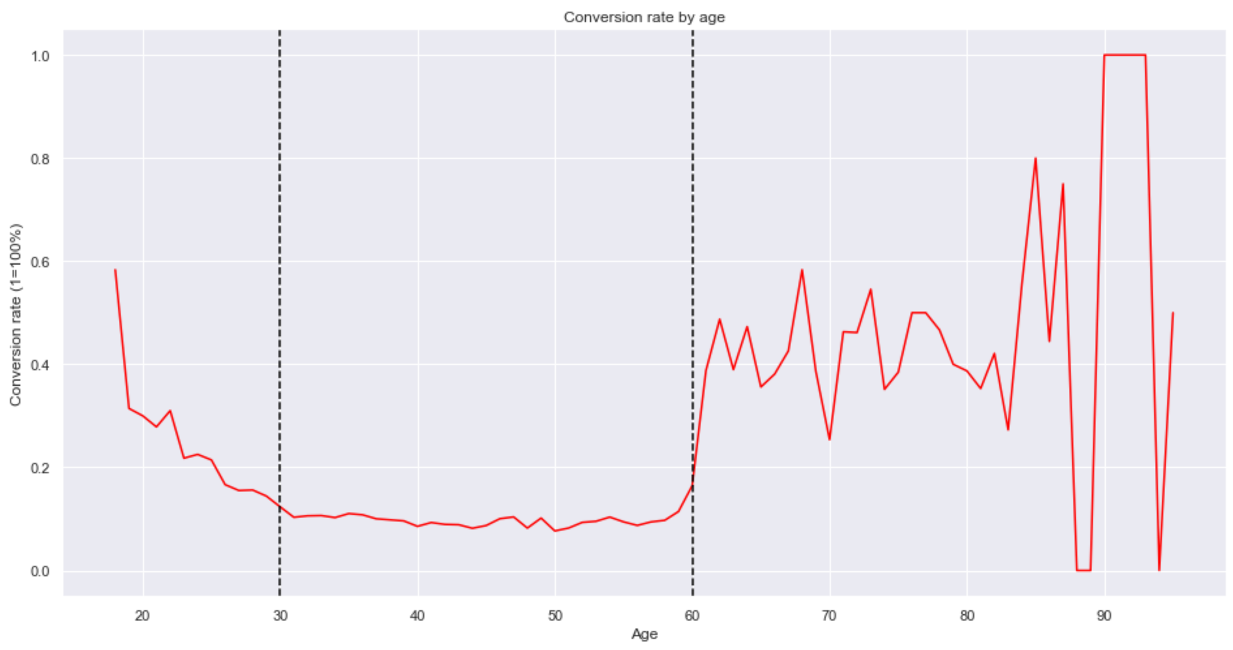


March and December seem to be the best months to hold campaigns in as they have the best results. Campaigning can be relaxed I the months of May, June and November, as not many customers seem to have a positive response to the campaigning.

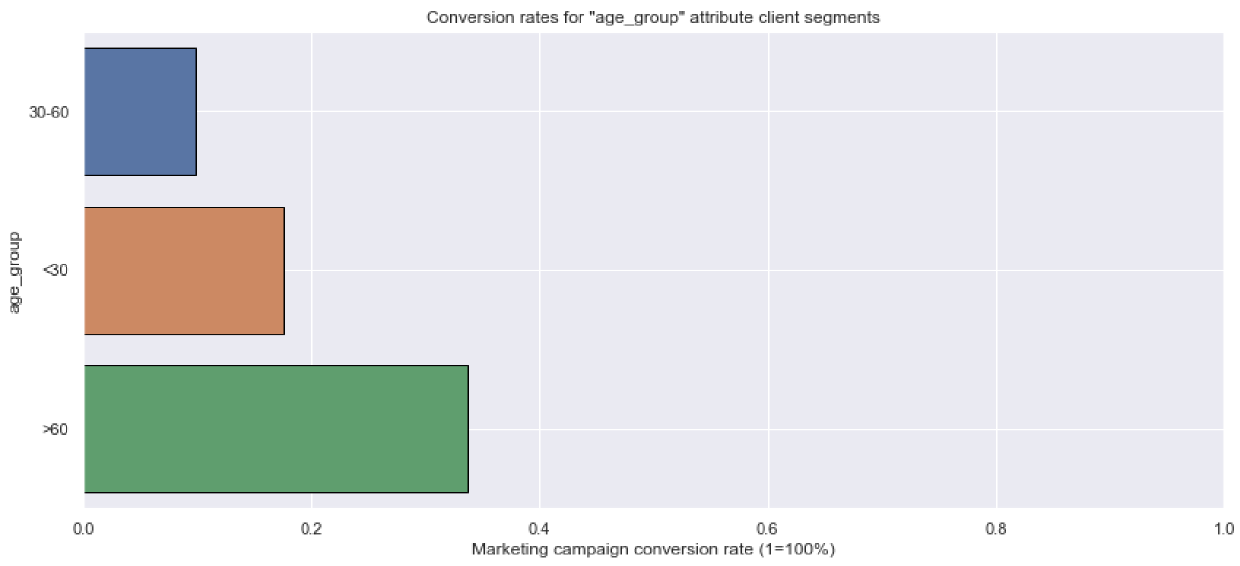


	age	balance	campaign	pdays	previous	y
age	1	0.098	0.0048	-0.024	0.0013	0.025
balance	0.098	1	-0.015	0.0034	0.017	0.053
campaign	0.0048	-0.015	1	-0.089	-0.033	-0.073
pdays	-0.024	0.0034	-0.089	1	0.45	0.1
previous	0.0013	0.017	-0.033	0.45	1	0.093
y	0.025	0.053	-0.073	0.1	0.093	1

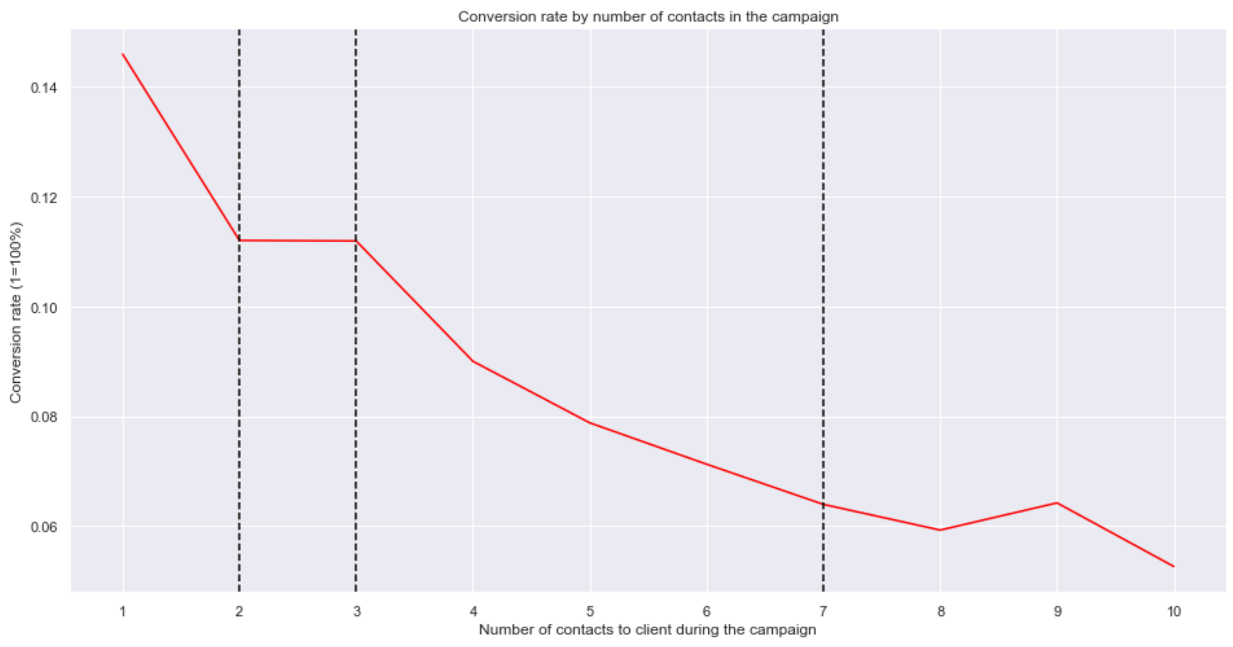
None of the features seem to hold high correlation, thereby removing the need to do dimension reduction.



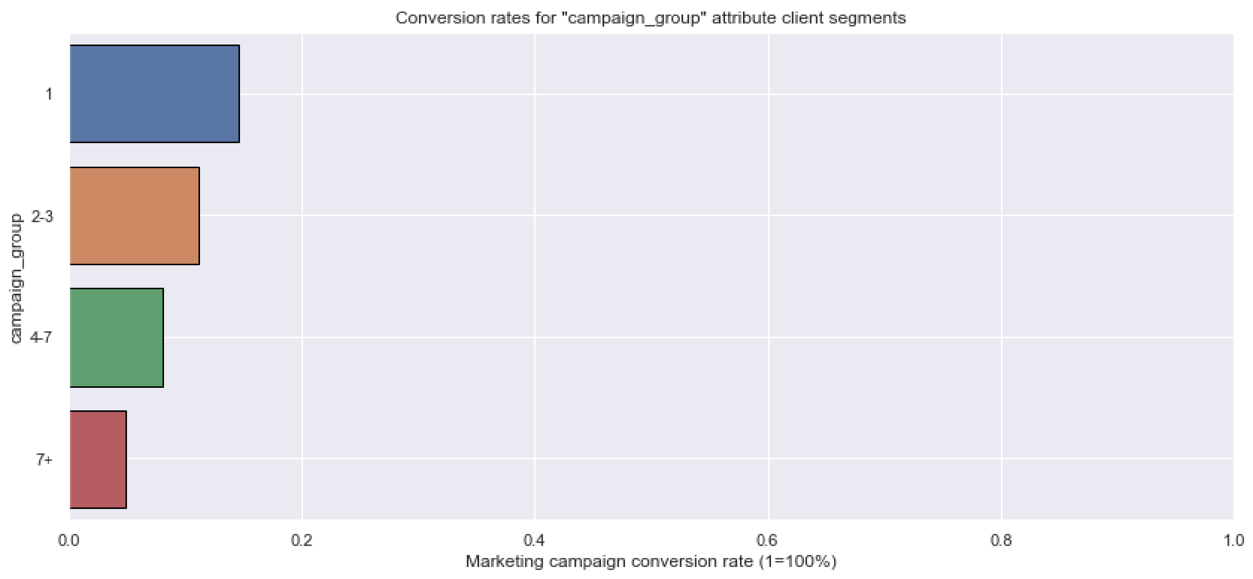
People in ages below 30 and above 60 tend to show more interest in opening accounts as these are the age-groups where people do not have exorbitant expenses and tend to save more.



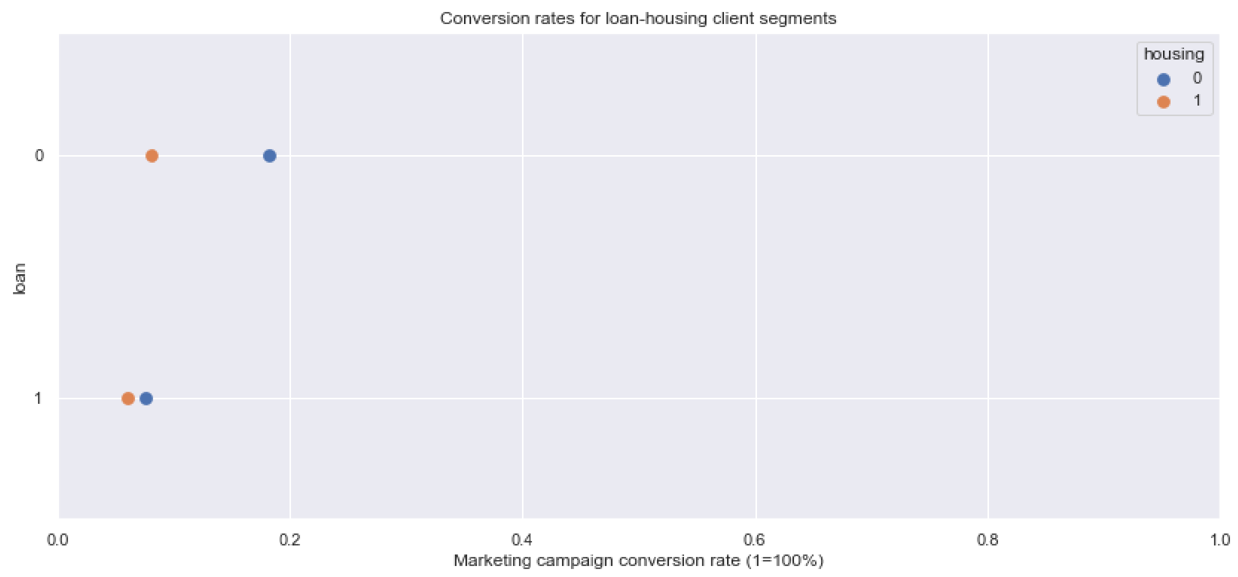
This is another representation of the line chart. It also depicts the age group between 30 and 60 showing the least interest in opening an account.



The relationship between the conversion rate of a person opening an account in response to the number of calls made to him in the campaign is shown above. The greater number of calls made, the less likely it is that the person would open an account.



The relationship between the number of calls made to a potential customer and his/her conversion rate is depicted in the bar chart above. The campaign is most successful if only a single call is made.



There isn't much of a difference in the chance of converting it or not if a customer has a housing property or not according to the point plot above.

RESULTS & OBSERVATIONS:

Logistic Regression:

<u>Model Description:</u>

For the Logistic Regression model without SMOTE, the parameters we used were:

i) Learning rate of 0.000001
ii) Tolerance of 0.0001

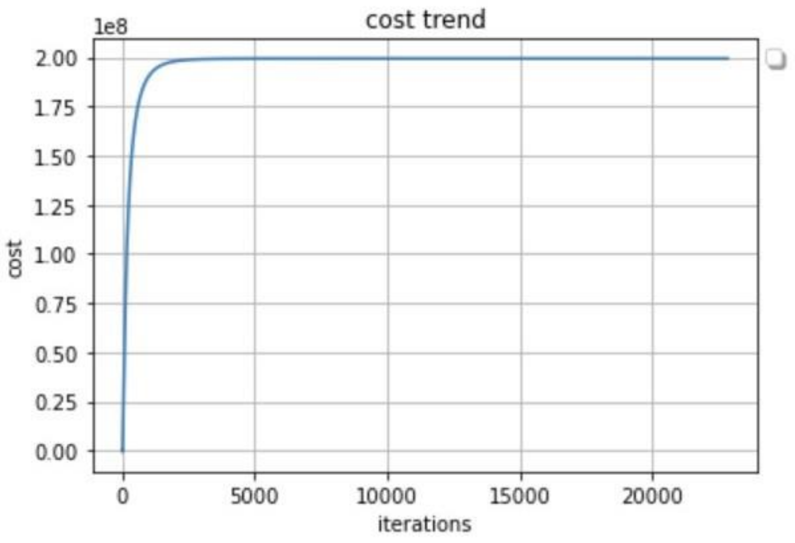
If we keep decreasing the learning rate the model takes longer to learn. Increasing the learning rate might overshoot the optimal solution.

1) Without SMOTE:

<u>Result:</u>

```
 46%|██████████████████████████████████
No handles with labels found to put in legend.
```

```
The Model Stopped - No Further Improvement
```



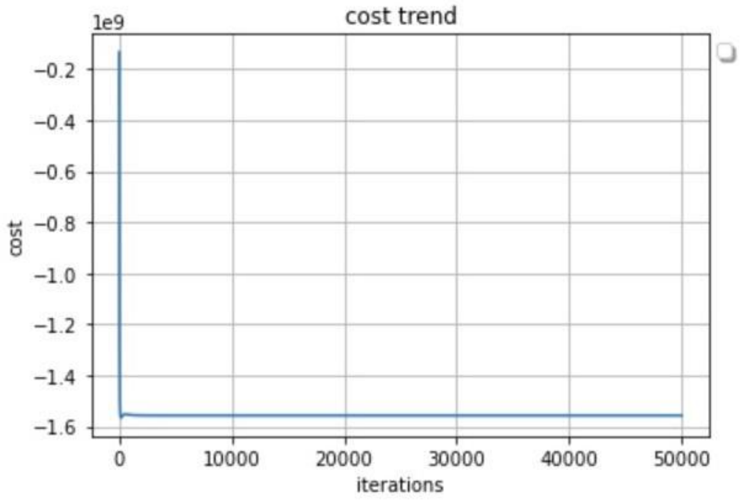
```
This is for y_train
Training Accuracy: 0.8923910639239107
Training Precision: 0.6317016317016317
Training Recall: 0.19215315528243915
Training f1score: 0.2946719826023922
This is for y_test
Training Accuracy: 0.8914077186774301
Training Precision: 0.6217948717948718
Training Recall: 0.1833648393194707
Training f1score: 0.2832116788321168
```

Without synthetic oversampling, logistic regression attains an accuracy of 0.89 over the output variable. This is a very good model.

2) With SMOTE:

<u>Result:</u>

```
100%|████████████████████████████████████████████████████████████████████████|
No handles with labels found to put in legend.
```



```
This is for y_train
Training Accuracy: 0.7161842229868367
Training Precision: 0.7446154945172697
Training Recall: 0.6580700148363269
Training f1score: 0.698672833299819
This is for y_test
Training Accuracy: 0.7601459692579896
Training Precision: 0.2760983474405482
Training Recall: 0.6474480151228733
Training f1score: 0.38711500423848544
```

Using oversampling, we attain a accuracy of 0.76 for the out variable with logistic regression. The F1 score is 0.387. This gives a reasonable result.

This proves that SMOTE isn't required while doing logistic regression.

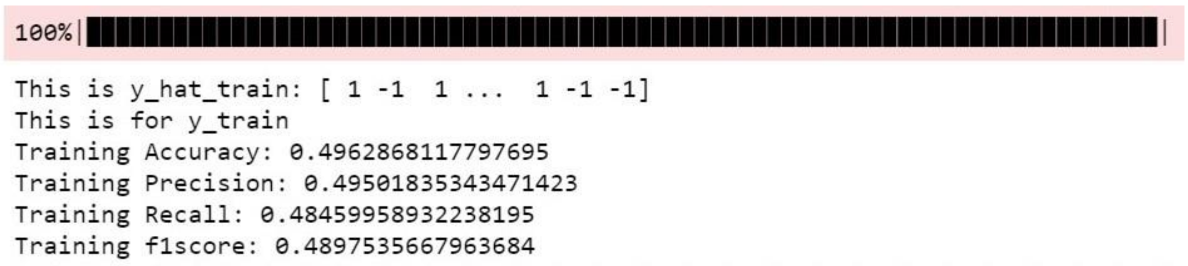
SVM:

Model Description:

SVM with the kernel function allows us to classify data points when the separator is non-linear. The function converts a linear line into a higher dimensional function.

Result:

We use the SVM with RBF kernel to classify data points into the two out variables.



```
100%|████████████████████████████████████████████████████████████████████████|

This is y_hat_train: [ 1 -1  1 ...  1 -1 -1]
This is for y_train
Training Accuracy: 0.4962868117797695
Training Precision: 0.49501835343471423
Training Recall: 0.48459958932238195
Training f1score: 0.4897535667963684
```

We attain an accuracy of 0.49 with this model along with an F1 score of 0.48. This isn't the best model to use to classify this data for this use-case.

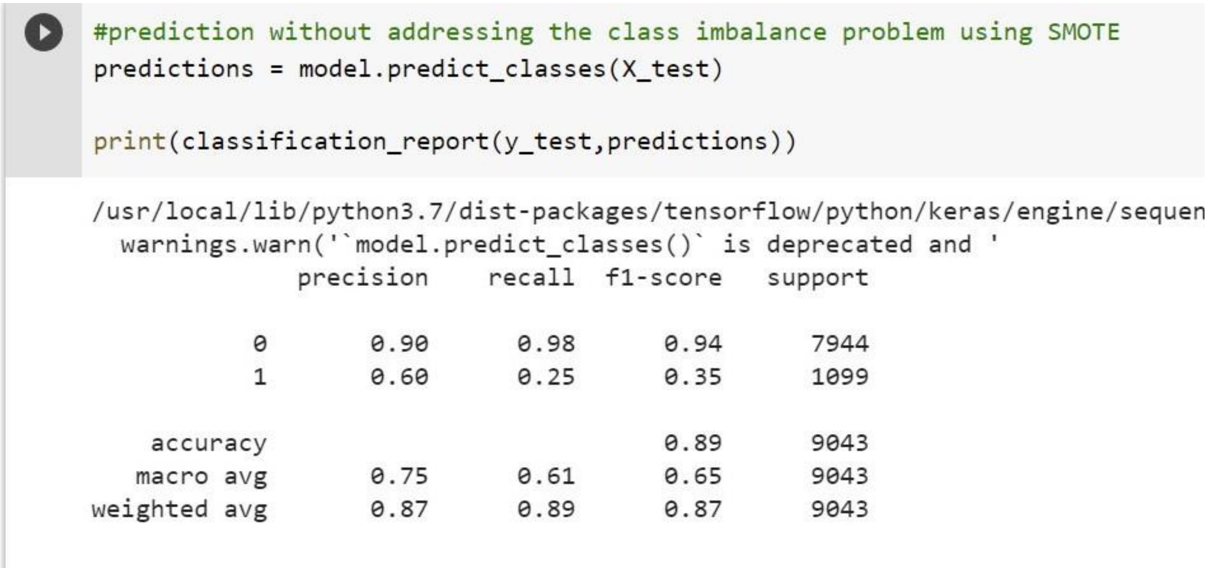
Neural Networks:

Model Description:

Early stopping is used to automatically detect overfitting and 100 epochs. The batch size is 100. A lower batch size allows us to come out of a local minima and a larger batch size causes the solution to oscillate.

Results:

1) Without SMOTE:

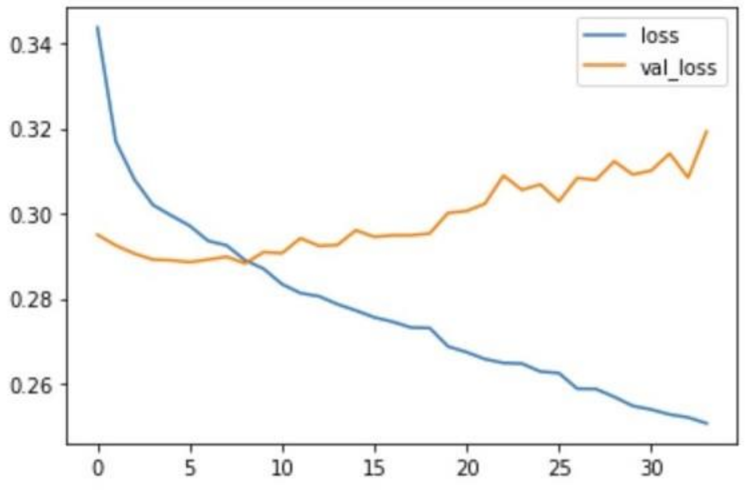


```
#prediction without addressing the class imbalance problem using SMOTE
predictions = model.predict_classes(X_test)

print(classification_report(y_test,predictions))
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequen
  warnings.warn('`model.predict_classes()` is deprecated and '
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      7944
           1       0.60      0.25      0.35      1099

    accuracy                           0.89      9043
   macro avg       0.75      0.61      0.65      9043
weighted avg       0.87      0.89      0.87      9043
```

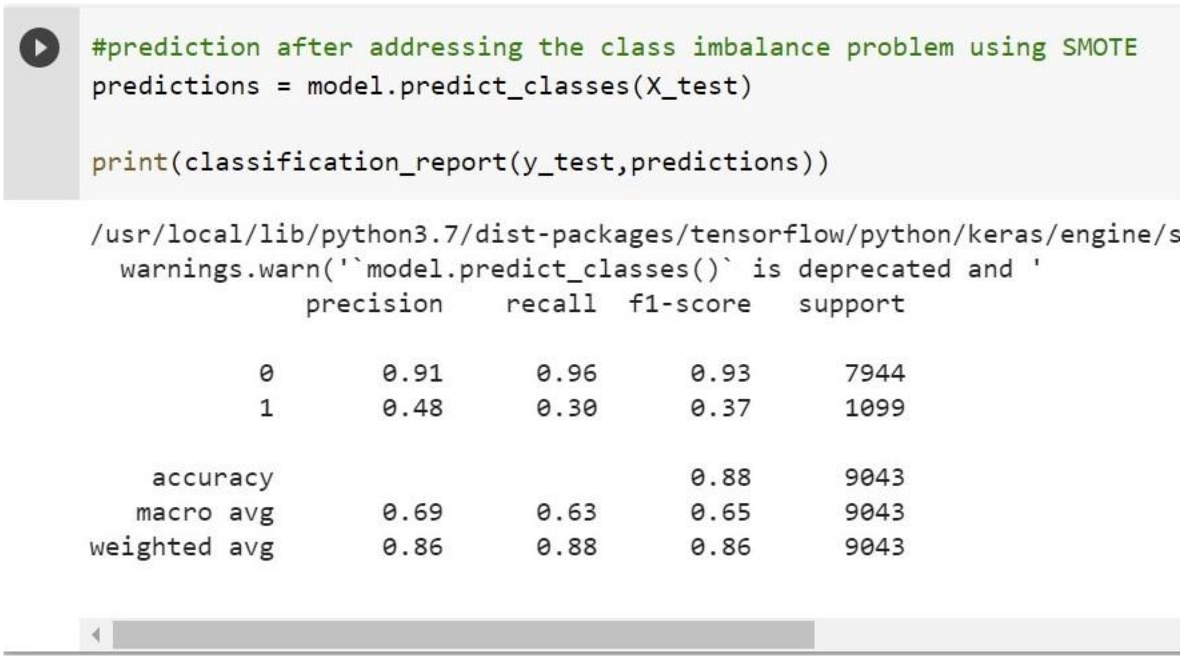
Without oversampling, we get an accuracy of 89%. This tells us that our model is doing well in predicting the labels for each data point. But since we have a class imbalance, we cannot completely rely on accuracy. The F1 score is a better metric to optimize. The F1 score is 0.94. So, we have a good model.

```
In [ ]:  losses = pd.DataFrame(model.history.history)

         losses[['loss','val_loss']].plot()

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1db61c2710>
```



The training loss reduces as the validation loss increases.

2) With SMOTE:

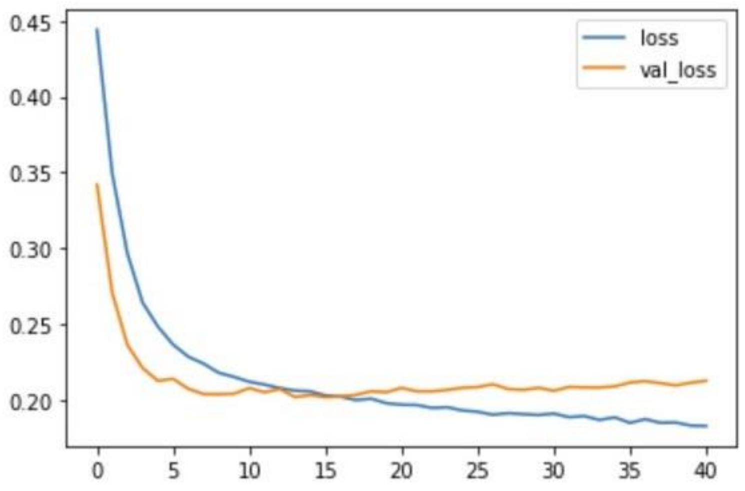


```
#prediction after addressing the class imbalance problem using SMOTE
predictions = model.predict_classes(X_test)

print(classification_report(y_test,predictions))
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/s
  warnings.warn('`model.predict_classes()` is deprecated and '
              precision    recall  f1-score   support

           0       0.91      0.96      0.93      7944
           1       0.48      0.30      0.37      1099

    accuracy                           0.88      9043
   macro avg       0.69      0.63      0.65      9043
weighted avg       0.86      0.88      0.86      9043
```

Using oversampling, we don't get a big difference in the accuracies or the F1 score. This means that oversampling isn't required for this sample of data.

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1db8247c50>
```



The training loss reduces to an optimum value while the validation loss remains constant. This infers that this is a better model without overfitting or underfitting.

DISCUSSION:

Throughout the course of the project, we incorporated techniques taught in class. For our baseline model, we used a logistic regression model. This model is a linear classifier. As the data we used isn't linearly separable, we saw that the baseline model performed the worst.

The second classical Machine Learning model we implemented was SVM using RBF Kernel. As this model is a non-linear classifier, it performed better as compared to the logistic regression model.

The final model we implemented is a feed-forward neural network using the Tensorflow and Keras packages. Through the implementation process, by playing around with the hyperparameter values, we found out that this model performs the best.