MICRO-CREDIT DEFAULTER MODEL

Submitted by:

SARANSH SHARDA

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

INTRODUCTION

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients. They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

Conceptual Background of the Domain Problem

Many donors, experts, and microfinance institutions (MFI) have become convinced that using mobile financial services (MFS) is more convenient and efficient, and less costly, than the traditional high-touch model for delivering microfinance services. MFS becomes especially useful when targeting the unbanked poor living in remote areas. The implementation of MFS, though, has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

One of our Client in Telecom collaborates with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be delinquent if he deviates from the path of paying back the loaned amount within 5 days.

Review of Literature

A machine learning model is created which can predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan and after the research it is shown that out of different models used, RANDOM FOREST will have minimal errors with maximum accuracy.

Motivation for the Problem Undertaken

To Build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan..

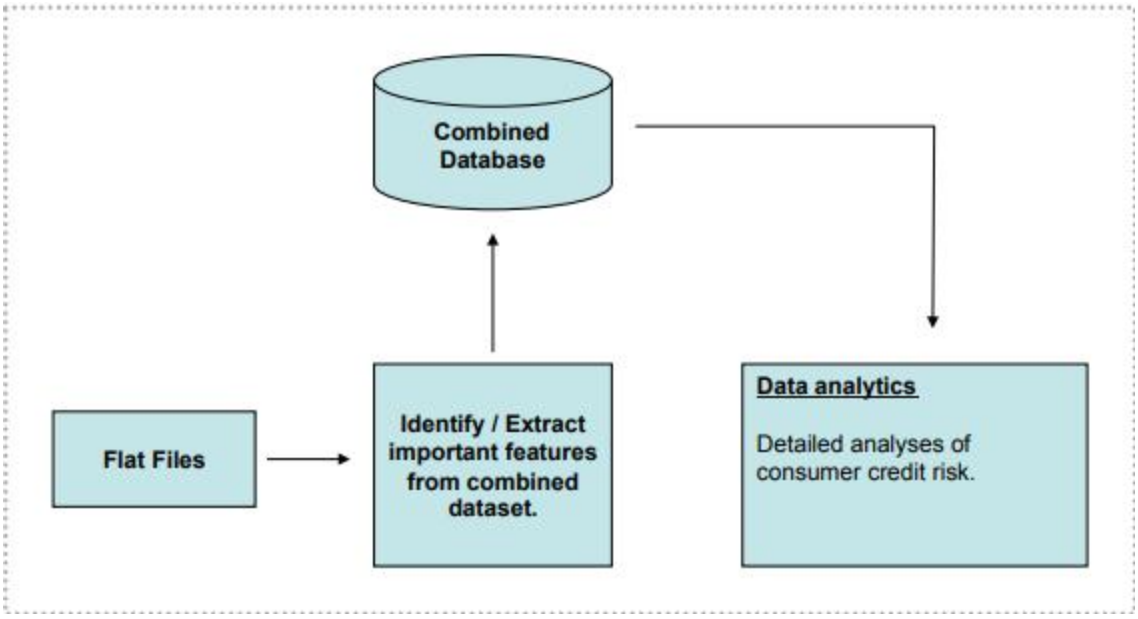
Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

Different type of classification modelling is performed to find the best predictive model for the micro-credit defaulter data like Random forest ,Logistic Regression ,GAUSSIAN NAIVEBAYES ,Decision Tree, Support Vector Machine

Data Sources and their formats

The raw data was provided to us in flat-file format. A set of programs were then used to pre-process the data, producing higher-level variables or “feature vectors” and time-aggregated to yield monthly observations that were stored in a database see Figure below. The following is a description of each of these components.



Data Preprocessing

- STEPS-

1. Dropped un necessary columns as “pcircle,msisdn,pdate”.

2 .As the data was Imbalanced so scaling over data was done.

3. Imported normalizer and normalise the data.

4. Imported stats from “scipy” and refined the outliers have z score more than 3.

Data Inputs- Logic- Output Relationships

It is shown that Via box plots, the data was completely co related with the “label” (target column), as it is completely showing how much defaulters carry average balance and data in last 30 - 90 days, another way to find relationship is the syntax “df.corr()” will show the relationships between all.

Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

Lists of packages and libraries used in modelling --

1.from sklearn.metrics import f1_score,roc_curve, classification_report, confusion_matrix

2. from sklearn.linear_model import LogisticRegression

3. from sklearn import metrics

4. from sklearn.ensemble import RandomForestClassifier

5. from sklearn import model_selection

6. import matplotlib.pyplot as plt

7. from sklearn.preprocessing import LabelEncoder

8. from sklearn.naive_bayes import GaussianNB

9. from sklearn.tree import DecisionTreeClassifier

10. from sklearn.linear_model import LogisticRegression

11. from sklearn.svm import SVC

12. from sklearn.metrics import accuracy_score, recall_score, precision_score, precision_recall_curve

13.from sklearn.model_selection import train_test_split

14. import pandas as pd

15.import numpy as np

16. import seaborn as sns

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

After train test splitting the data, the basic approach followed was checking the difference in confusion matrix and comparing f1 score between models with comparing AUC-ROC curve

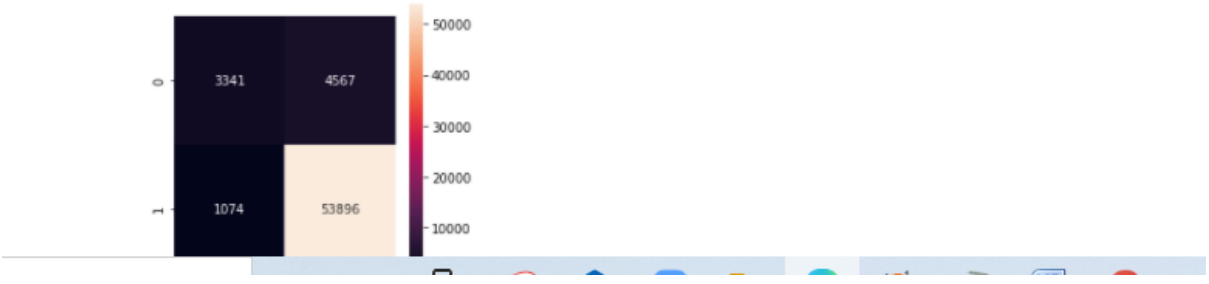
EXAMPLE-

In [201]:
```
#RANDOM FOREST CLASSIFIER
rf =RandomForestClassifier()
rf.fit(x_train,y_train)
acc,rocauc,testpred_rf  = evaluate_model(rf)
print('Random Forest...')
Y_RFpred=rf.predict(x_test)
print(classification_report(Y_RFpred,y_test))
```

```
Random Forest...
              precision    recall  f1-score   support

           0       0.42      0.76      0.54      4415
           1       0.98      0.92      0.95     58463

    accuracy                           0.91     62878
   macro avg       0.70      0.84      0.75     62878
weighted avg       0.94      0.91      0.92     62878
```

In [202]:
```
cm_RF = confusion_matrix(y_test, Y_RFpred)
_,ax = plt.subplots(figsize=(4,4))
sns.heatmap(cm_RF,annot=True,fmt="d")
ax.set_ylim(2,-0.1)
```

Out[202]: (2.0, -0.1)



Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

all the algorithms used for the training and testing-

Random forest

Logistic Regression

GAUSSIAN NAIVEBAYES

Decision Tree

Support Vector Machine

Run and Evaluate selected models

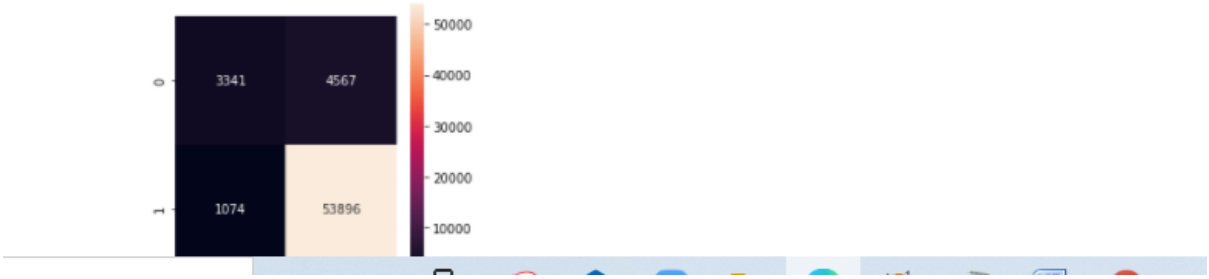
Random forest

In [201]:
```
#RANDOM FOREST CLASSIFIER
rf =RandomForestClassifier()
rf.fit(x_train,y_train)
acc,rocauc,testpred_rf  = evaluate_model(rf)
print('Random Forest...')
Y_RFpred=rf.predict(x_test)
print(classification_report(Y_RFpred,y_test))
```

```
Random Forest...
              precision    recall  f1-score   support

           0       0.42      0.76      0.54      4415
           1       0.98      0.92      0.95     58463

    accuracy                           0.91     62878
   macro avg       0.70      0.84      0.75     62878
weighted avg       0.94      0.91      0.92     62878
```

In [202]:
```
cm_RF = confusion_matrix(y_test, Y_RFpred)
_,ax = plt.subplots(figsize=(4,4))
sns.heatmap(cm_RF,annot=True,fmt="d")
ax.set_ylim(2,-0.1)
```

Out[202]: (2.0, -0.1)



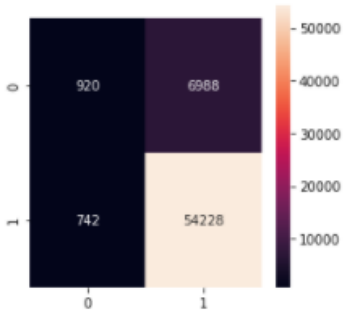
Logistic Regression

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Logistic Regression...
              precision    recall  f1-score   support

           0       0.12      0.55      0.19      1662
           1       0.99      0.89      0.93     61216

    accuracy                           0.88     62878
   macro avg       0.55      0.72      0.56     62878
weighted avg       0.96      0.88      0.91     62878
```

```
cm_LR = confusion_matrix(y_test, Y_LRpred)
_,ax = plt.subplots(figsize=(4,4))
sns.heatmap(cm_LR,annot=True,fmt="d")
ax.set_ylim(2,-0.1)
```

```
(2.0, -0.1)
```



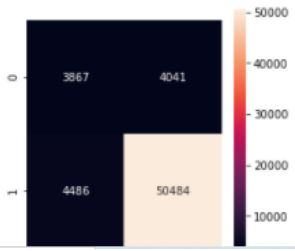
GAUSSIAN NAIVEBAYES

In [207]:
```
#GAUSSIAN NAIVEBAYES
from sklearn.naive_bayes import GaussianNB
gnb =DecisionTreeClassifier()
gnb.fit(x_train,y_train)
acc,rocauc,testpred_gnb = evaluate_model(gnb)
print('Gaussian Navie Bayes...')
Y_GNBpred=gnb.predict(x_test)
print(classification_report(Y_GNBpred,y_test))
```

```
Gaussian Navie Bayes...
              precision    recall  f1-score   support

           0       0.49      0.46      0.48      8353
           1       0.92      0.93      0.92     54525

    accuracy                           0.86     62878
   macro avg       0.70      0.69      0.70     62878
weighted avg       0.86      0.86      0.86     62878
```

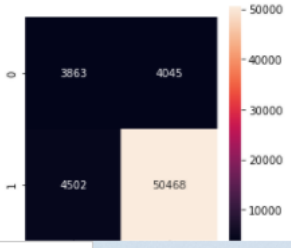
In [208]:
```
cm_GNB = confusion_matrix(y_test, Y_GNBpred)
_,ax = plt.subplots(figsize=(4,4))
sns.heatmap(cm_GNB,annot=True,fmt="d")
ax.set_ylim(2,-0.1)
```

Out[208]:
```
(2.0, -0.1)
```



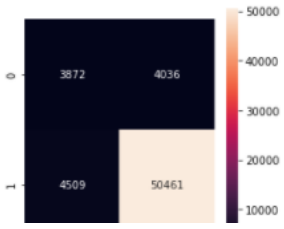
Decision Tree

```
In [205]: from sklearn.tree import DecisionTreeClassifier
          dt =DecisionTreeClassifier()
          dt.fit(x_train,y_train)
          acc,rocauc,testpred_dt = evaluate_model(dt)
          print('Decision Tree Classifier...')
          Y_DTpred=dt.predict(x_test)
          print(classification_report(Y_DTpred,y_test))

          Decision Tree Classifier...
                        precision    recall  f1-score   support

                     0       0.49      0.46      0.47      8365
                     1       0.92      0.93      0.92     54513

              accuracy                           0.86     62878
             macro avg       0.70      0.69      0.70     62878
          weighted avg       0.86      0.86      0.86     62878

In [206]: cm_DT = confusion_matrix(y_test, Y_DTpred)
          _,ax = plt.subplots(figsize=(4,4))
          sns.heatmap(cm_DT,annot=True,fmt="d")
          ax.set_ylim(2,-0.1)

Out[206]: (2.0, -0.1)
```



Support Vector

```
In [209]: #SUPPORT VECTOR MACHINE
          from sklearn.svm import SVC
          svc =DecisionTreeClassifier()
          svc.fit(x_train,y_train)
          acc,rocauc,testpred_svc = evaluate_model(svc)
          print('Support vector classifier...')
          Y_SVCpred=svc.predict(x_test)
          print(classification_report(Y_SVCpred,y_test))

          Support vector classifier...
                        precision    recall  f1-score   support

                     0       0.49      0.46      0.48      8381
                     1       0.92      0.93      0.92     54497

              accuracy                           0.86     62878
             macro avg       0.70      0.69      0.70     62878
          weighted avg       0.86      0.86      0.86     62878

In [210]: cm_SVC = confusion_matrix(y_test, Y_SVCpred)
          _,ax = plt.subplots(figsize=(4,4))
          sns.heatmap(cm_SVC,annot=True,fmt="d")
          ax.set_ylim(2,-0.1)

Out[210]: (2.0, -0.1)
```



(Out of all, Random forest model has an Accuracy of 0.91 and AUC_ROC score of 0.70. we also have....
True Negative = 3341 False Positive = 1074

False Negative = 4567 True Positive = 53896 Also, we had low Type-1 Error as FP is less.)

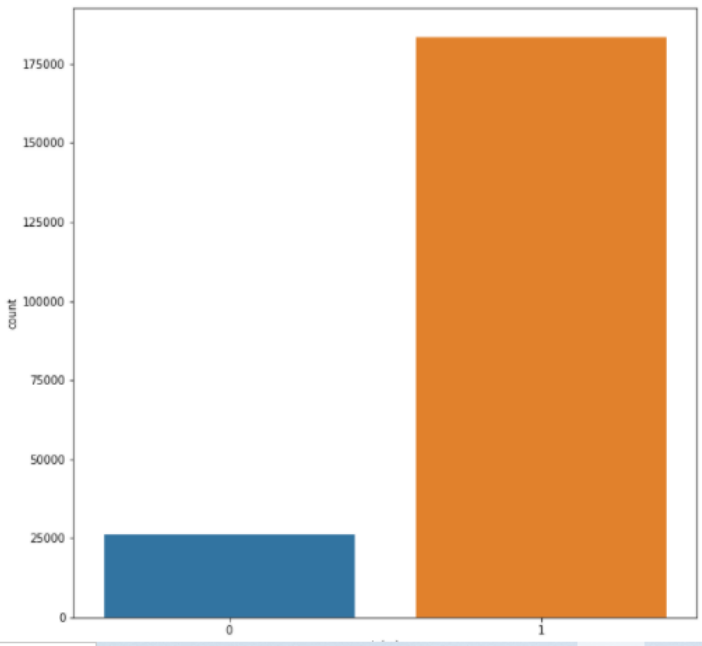
Key Metrics for success in solving problem under consideration

Accuracy score, recall score, precision score, confusion matrix and classification reports were generated compared for solving model.

- Visualizations

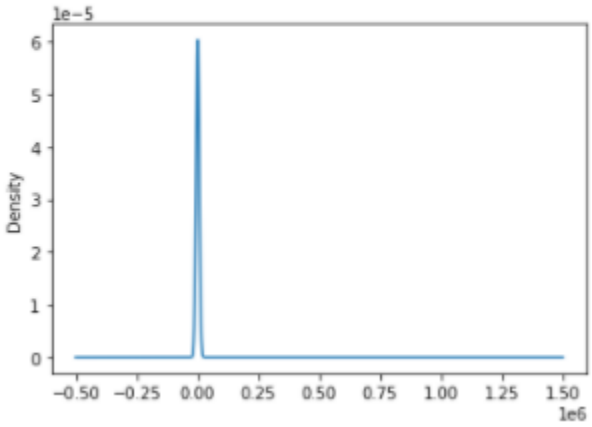
Target column (label count plot)

```
In [100]: plt.figure(figsize=(10,10))
          sns.countplot(x="label", data=df)
          plt.show()
```



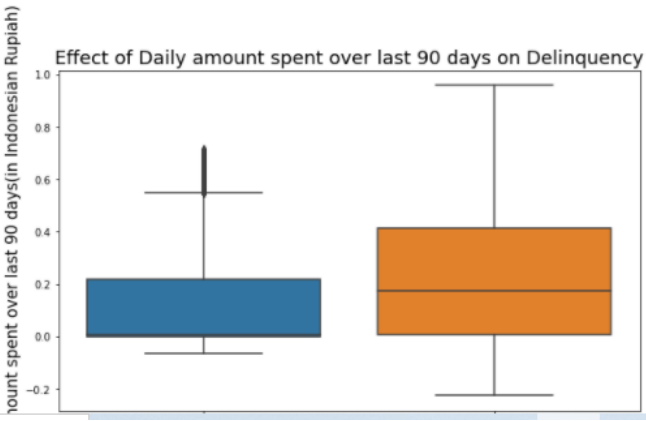
Column “aon” plot to check the density of values which shows that it lies near 0 only

```
n [105]: df.aon.plot(kind='density')
ut[105]: <AxesSubplot:ylabel='Density'>
```



Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)

```
In [120]: #Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
plt.figure(figsize=(10,6))
ax = sns.boxplot(y, x['daily_decr90'])
ax.set_title('Effect of Daily amount spent over last 90 days on Delinquency', fontsize=18)
ax.set_ylabel('Daily amount spent over last 90 days(in Indonesian Rupiah)', fontsize = 15)
ax.set_xlabel('Delinquency', fontsize = 15)

C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(

Out[120]: Text(0.5, 0, 'Delinquency')
```

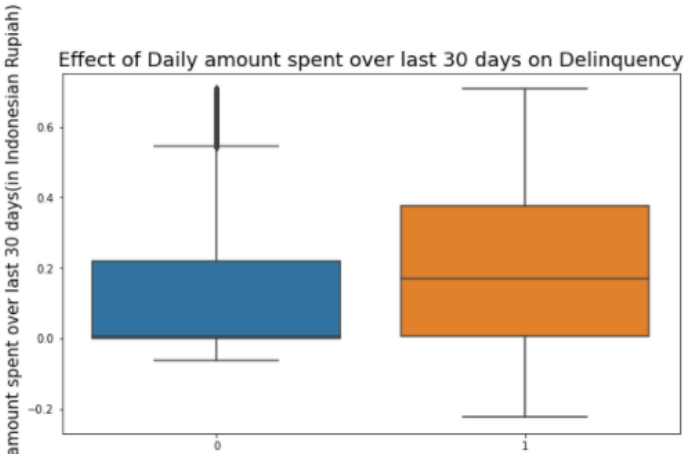


Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)

```
In [121]: #Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
plt.figure(figsize=(10,6))
ax = sns.boxplot(y, x['daily_decr30'])
ax.set_title('Effect of Daily amount spent over last 30 days on Delinquency', fontsize=18)
ax.set_ylabel('Daily amount spent over last 30 days(in Indonesian Rupiah)', fontsize = 15)
ax.set_xlabel('Delinquency', fontsize = 15)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[121]: Text(0.5, 0, 'Delinquency')
```

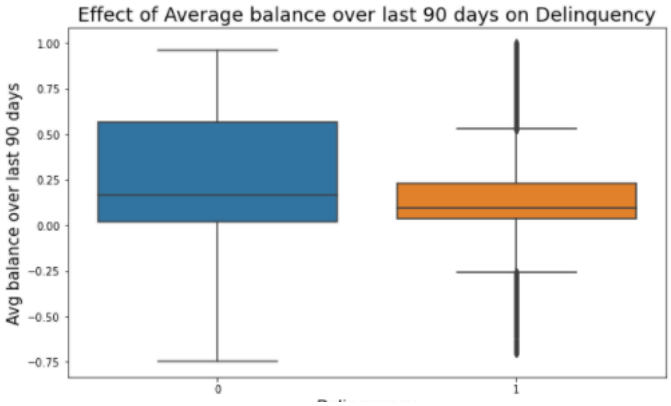


Average main account balance over last 90 days

```
In [122]: #Average main account balance over last 90 days
plt.figure(figsize=(10,6))
ax = sns.boxplot(y, x['rental90'])
ax.set_title('Effect of Average balance over last 90 days on Delinquency', fontsize=18)
ax.set_ylabel('Avg balance over last 90 days', fontsize = 15)
ax.set_xlabel('Delinquency', fontsize = 15)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[122]: Text(0.5, 0, 'Delinquency')
```

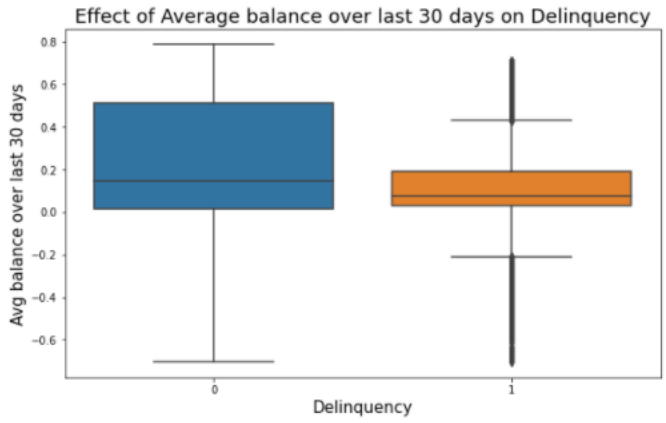


Average main account balance over last 30 days

```
In [123]: #Average main account balance over last 30 days
          plt.figure(figsize=(10,6))
          ax = sns.boxplot(y, x['rental30'])
          ax.set_title('Effect of Average balance over last 30 days on Delinquency', fontsize=18)
          ax.set_ylabel('Avg balance over last 30 days', fontsize = 15)
          ax.set_xlabel('Delinquency', fontsize = 15)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[123]: Text(0.5, 0, 'Delinquency')
```



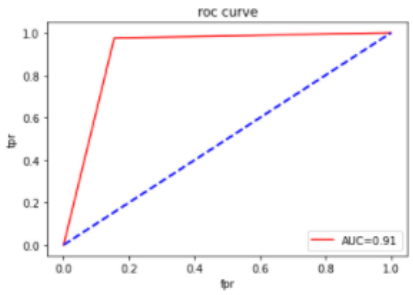
Interpretation of the Results

After all data cleaning and processing and scaling the data, I summarize that we can go with the most effective classification modelling technique for better result showing high accuracy and that is RANDOM FOREST.

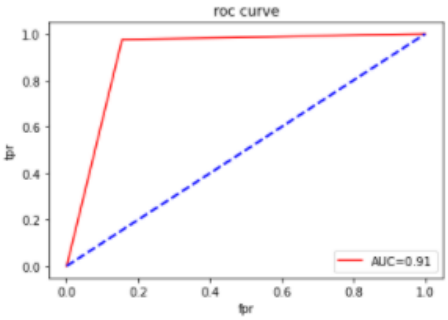
CONCLUSION

- Key Findings and Conclusions of the Study

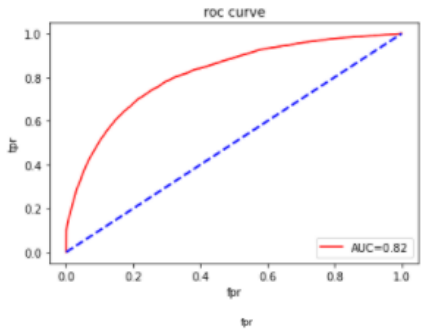
In [213]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
preds=dt.predict_proba(x)[:,1]
fpr,tpr,thershold=roc_curve(y,preds,drop_intermediate=False)
roc_auc=auc(fpr,tpr)
plt.plot(fpr,tpr,'r',label='AUC=%0.2f'% roc_auc)
plt.plot([0,1],[0,1],color='blue',lw=2,linestyle='--')
plt.legend(loc='lower right')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('roc curve')
plt.show()
```



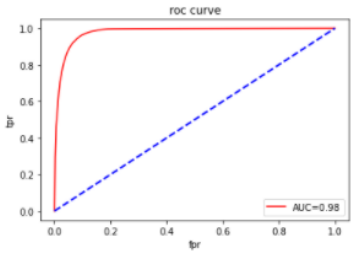
In [214]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
preds=gnb.predict_proba(x)[:,1]
fpr,tpr,thershold=roc_curve(y,preds,drop_intermediate=False)
roc_auc=auc(fpr,tpr)
plt.plot(fpr,tpr,'r',label='AUC=%0.2f'% roc_auc)
plt.plot([0,1],[0,1],color='blue',lw=2,linestyle='--')
plt.legend(loc='lower right')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('roc curve')
plt.show()
```



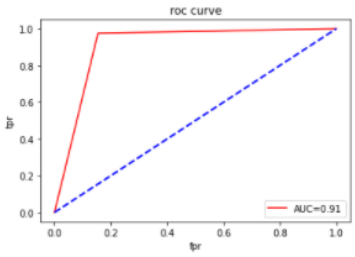
In [211]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
preds=lr.predict_proba(x)[:,1]
fpr,tpr,thershold=roc_curve(y,preds,drop_intermediate=False)
roc_auc=auc(fpr,tpr)
plt.plot(fpr,tpr,'r',label='AUC=%0.2f'% roc_auc)
plt.plot([0,1],[0,1],color='blue',lw=2,linestyle='--')
plt.legend(loc='lower right')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('roc curve')
plt.show()
```



In [212]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
preds=rf.predict_proba(x)[:,1]
fpr,tpr,thershold=roc_curve(y,preds,drop_intermediate=False)
roc_auc=auc(fpr,tpr)
plt.plot(fpr,tpr,'r',label='AUC=%0.2f'% roc_auc)
plt.plot([0,1],[0,1],color='blue',lw=2,linestyle='--')
plt.legend(loc='lower right')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('roc curve')
plt.show()
```



In [215]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
preds=svc.predict_proba(x)[:,1]
fpr,tpr,thershold=roc_curve(y,preds,drop_intermediate=False)
roc_auc=auc(fpr,tpr)
plt.plot(fpr,tpr,'r',label='AUC=%0.2f'% roc_auc)
plt.plot([0,1],[0,1],color='blue',lw=2,linestyle='--')
plt.legend(loc='lower right')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('roc curve')
plt.show()
```

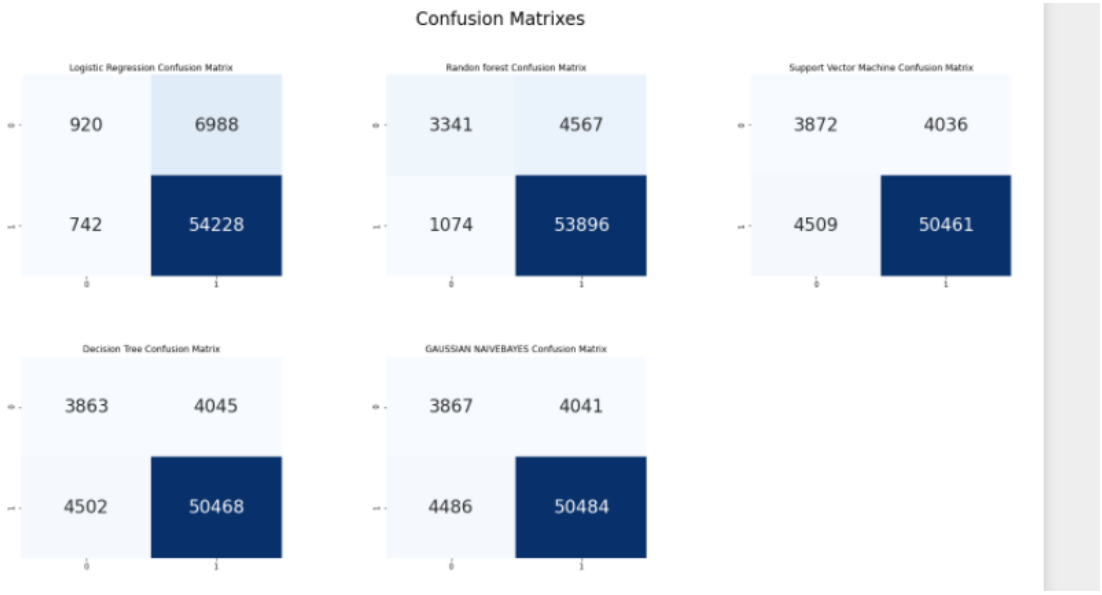


Out of all of the roc curves above, we can conclude that random forest have better area under curve value.

We can finally conclude that—

"The final model has an Accuracy of 0.91 and AUC_ROC score of 0.70. we also have.... True Negative = 3341 False Positive = 1074 False Negative = 4567 True Positive = 53896 Also, we had low Type-1 Error as FP is less."

CONFUSION MATRIX for all models –



- Learning Outcomes of the Study in respect of Data Science

What I learnt during my whole project is that I should be aware of my data and how to handle my imbalanced data so that I can get perfect accuracy.

I should be aware of outliers and null values and how to over come these natural situations

I faced step wise problem, over how to write a particular code but after research I was able to get all confusion matrix along.

Limitations of this work and Scope for Future Work

Limitation – even after prediction the best model, confusion matrix still shows minimum error which can be solved with better solution but still the project can be handled for maximum revenue defaulter less data with the help of the model created above.

, the future scope? What all steps/techniques can be followed to further extend this study and improve the results.