MICRO CREDIT DEFAULTER PROJECT

Submitted by:

MOHAMMED MINHAJ

ACKNOWLEDGMENT

Wikipedia, Google, You tube and books about data science and machine learning. And the previous projects were also referred for completion of this project.

INTRODUCTION

- Business Problem Framing

Micro finance institution provides Micro finance services for low income families in rural areas. Which is very efficient and cost savings. As we know micro finance means small amount of loan. It is given by telecom companies for people. Giving credit money to mobile recharge and payback a particular time. When a customer not pay back the credit amount, he is non-defaulter otherwise defaulter. If company didn’t get the payback it will affect company’s income. So here we want to predict the customers who pay back the loan and company want to keep those customers.

- Conceptual Background of the Domain Problem

We want to find whether the user is non-defaulter or not that is the user have pay back the loan or not. Those who pay back the loan is consider as the non-defaulter. Our main objective is to find the user is defaulter or non-defaulter with the given data.

- Review of Literature

The main aim of this project is to find those customers who pay back the loan at a particular time. If any customer pay back the loan within five days that customer is non defaulter otherwise defaulter. So we are predicting defaulter and non-defaulter by analysing data related to this. For example, what is the balance of the account, is the account have sufficient balance to pay back, what is the balance of account at the time of taking Loan? Such type of data help us to predict our aim.

- Motivation for the Problem Undertaken

Objective behind this project is to giving small loan for low income families for their needs. The families who have no ability to have big loans and they can’t afford it and it may affect in their daily needs. The main motivation behind this project which is already mentioned. This all are for poor people, they can afford their needs in with this micro credit.

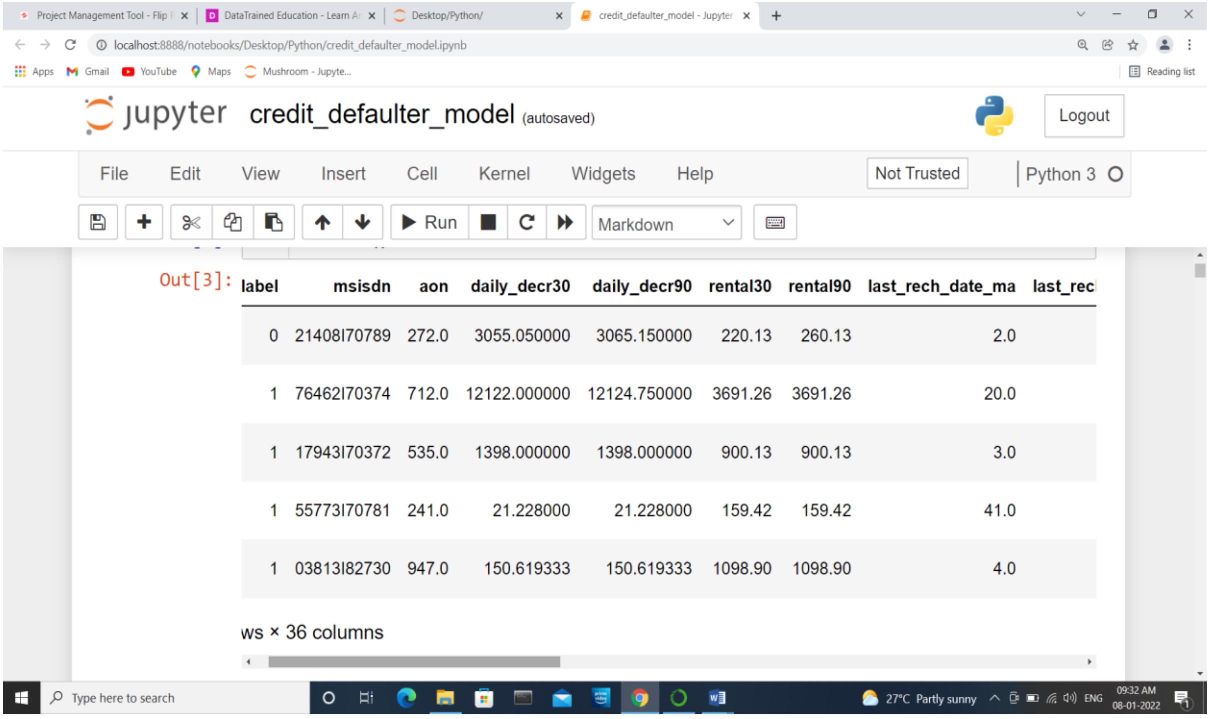
Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

Predictions are done by analysing the given data. The given data will be the past data. With this past data we want to predict future. So for analysing the past data we need to use some techniques for that. We want to relate each features, how they are related, how they related to the output. In this project the features are related, because the data are given in 30 days and 90 days. These features are related to each other and some of them were will give a high impact in output. We want to select those features and drop the features which is not give an impact for predictions.

- Data Sources and their formats

In this projects data are given in the format of currency (Indonesian rupiah). The loan amount, amount balance in account, recharges done in last 30 days and 90 days, number of recharges done. These all are the data sources for this project. The mobile number of the customers are given in the data sources. These are the data sources given for prediction.



Out[3]:

label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rec
0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	
1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	
1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	
1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	
1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	

ws × 36 columns

- Data Pre-processing Done

Here given data have no any missing values. But the majority of features contains value zero. Among 2 lakhs above data have 2 lakhs of zero value. Here I treated them as missing values. Treated those values using mean. Replaced those values by the mean of that features. And the data type of the feature mobile number is given as string. Here I encode this feature using label encoder. And features were scaled using standard scaler. When check for outliers, I can see the presence of outliers. When remove the outliers using zscore, there is 25% of data were loss. So I proceed with outliers because this much of losing data is not good for prediction. And the imbalance of class is present in data, balanced class using smote technique. These are the major data pre-processing done In this project. Here date is given as object, convert them into date time using pandas. Data are given from one region it may not affect the predictions so drop that feature from dataset.

- Data Inputs- Logic- Output Relationships

Here in this project most of the data are given in currency (Indonesian rupiah) when converting it into Indian currency one rupiah (Indonesian) is equal to 0.0052 Indian currency. So the majority of data is given about the account balance, maximum loan amount, how many times the recharges done Etc. are given in the data. Every features like recharges done, number of times recharge done Etc. are given in 30 days and 90 days. So every features given in 30 days are related 90 days. And these features gives an high impact on output. The loan amount given here 5 and 10 (Indonaesian rupiah) respectively. And want to payback 6 and 12 rupiah respectively in 5 days. The company want the customer who pay back correctly in date for investments. The features like the recharges done in 30 days, 90 days, maximum amount of loan taken, Balance of the account while taking the loan etc. these features affect high impact on the output. When customer have the balance in account can only pay back the loan.

- State the set of assumptions (if any) related to the problem under consideration

Customer who have balance in the account can only pay back the loan. Amount of recharge done in 30 days and 90 days, those customers can only payback the loan.

- Hardware and Software Requirements and Tools Used

Here when we look the memory usage of dataset is up to 60 MB. In this project I used pandas packages for analysing and visualizations of data. Numpy packages were used for arranging some range of numbers. Metrics like AUC ROC curve, accuracy score, confusion matrix, classification report. These are used for the performance of the model, which model have high accuracy and have good AUC curve will selected as best model. For performing algorithm we want to split into train and test data it is imported from the package model selection. From the same package Grid searchcv and cross validation score were imported for checking the best parameters for best algorithm and we can select best model by least difference in accuracy and CV score. Variance inflation factor is imported from stats model for checking the multi collinearity. Typical visualization tolls were used here Seaborn and Matplotlib. Standard scaler and label encoder is imported from pre-processing package for feature scaling and encoding respectively. From scipy stats Zscore were imported for removing outliers. Power transform used for removing skewness. Smote technique is used for balancing the class, it is imported from the package imblearn. Here we used 4 model decision tree, logistic regression, Bernoulli naïve bayes and random forest classifier. After comparing all model with metrics I choose random forest as my final model. At last file saved as pickle file.

Model/s Development and Evaluation

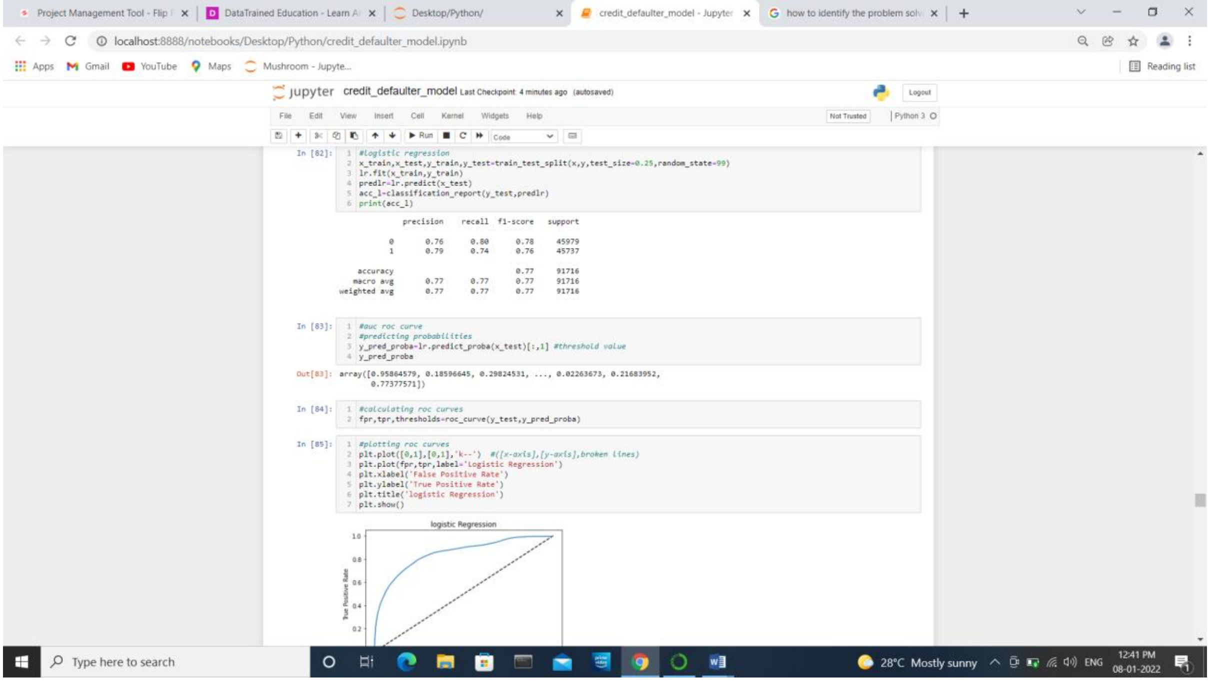
- Identification of possible problem-solving approaches (methods)

As our objective to predict the defaulter and non defaulter. For this we want some features which is helpful for our predictions. According to our project we need some features to know about the ability of customer to pay back the loan company wants the non defaulter customer for investing. Micro credit will benefits the company in so many ways, cost savings, efficient etc.

- Testing of Identified Approaches (Algorithms)
    - Logistic Regression
    - Decision Tree Classifier
    - Bernoulli NB
    - Random Forest Classifier

- Run and Evaluate selected models

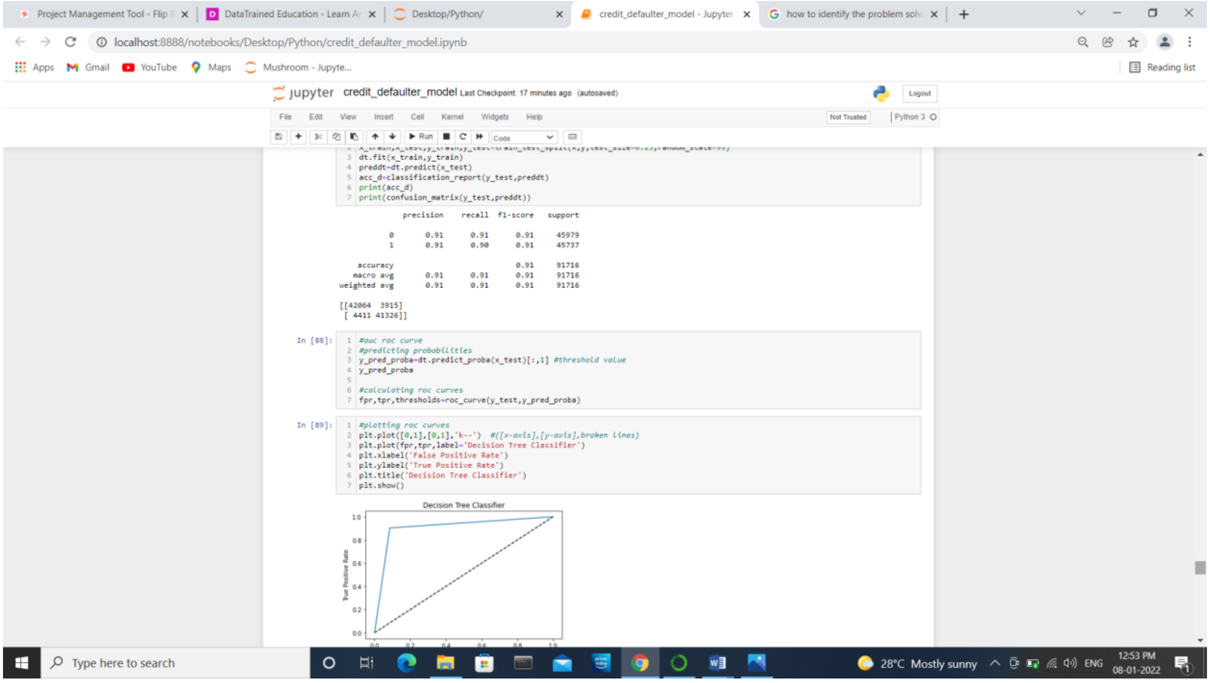
Logistic Regression


Project Management Tool - Flip
DataTrained Education - Learn
Desktop/Python/
credit_defaulter_model - Jupyter
how to identify the problem solv
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter credit_defaulter_model Last Checkpoint: 4 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Type here to search
28°C Mostly sunny
ENG
12:41 PM
08-01-2022

```
In [82]: #logistic regression
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=99)
         lr.fit(x_train,y_train)
         predlr=lr.predict(x_test)
         acc_l=classification_report(y_test,predlr)
         print(acc_l)

              precision    recall  f1-score   support

           0       0.76      0.80      0.78     45979
           1       0.79      0.74      0.76     45737

    accuracy                           0.77     91716
   macro avg       0.77      0.77      0.77     91716
weighted avg       0.77      0.77      0.77     91716

In [83]: #auc roc curve
         #predicting probabilities
         y_pred_proba=lr.predict_proba(x_test)[:,1] #threshold value
         y_pred_proba

Out[83]: array([0.95864579, 0.18596645, 0.29824531, ..., 0.02263673, 0.21683952,
                0.77377571])

In [84]: #calculating roc curves
         fpr,tpr,thresholds=roc_curve(y_test,y_pred_proba)

In [85]: #plotting roc curves
         plt.plot([0,1],[0,1],'k--')  #([x-axis],[y-axis],broken lines)
         plt.plot(fpr,tpr,label='Logistic Regression')
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('logistic Regression')
         plt.show()
```

We can observe the above picture that the accuracy for model logistic regression is 77%, precision and recall for class 0 is 76% and 80% respectively and for class 1 is 79% and 74%. When we are looking at AUC ROC curve it is not at all fine for logistic regression. Here we split 25% data for testing.

Decision Tree Classifier



```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=99)
dt.fit(x_train,y_train)
preddt=dt.predict(x_test)
acc_d=classification_report(y_test,preddt)
print(acc_d)
print(confusion_matrix(y_test,preddt))
```

```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     45979
           1       0.91      0.90      0.91     45737

    accuracy                           0.91     91716
   macro avg       0.91      0.91      0.91     91716
weighted avg       0.91      0.91      0.91     91716

[[42064  3915]
 [ 4411 41326]]
```

In [88]:

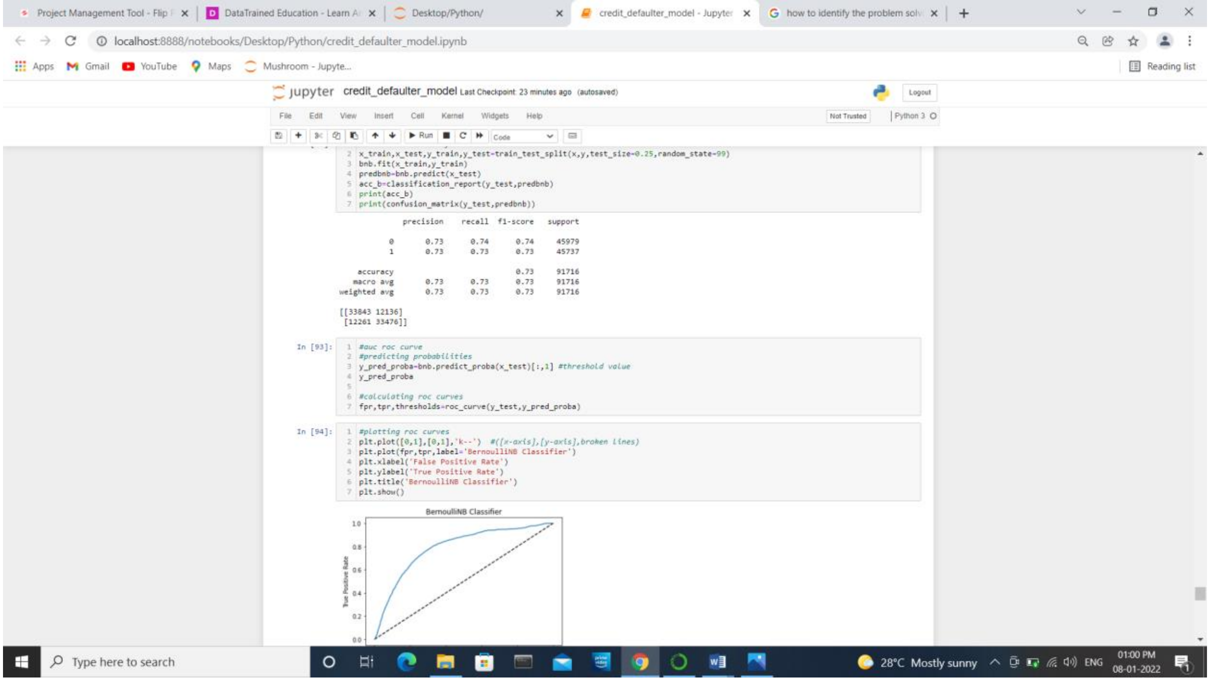
```
#auc roc curve
#predicting probabilities
y_pred_proba=dt.predict_proba(x_test)[:,1] #threshold value
y_pred_proba

#calculating roc curves
fpr,tpr,thresholds=roc_curve(y_test,y_pred_proba)
```

In [89]:

```
#plotting roc curves
plt.plot([0,1],[0,1],'k--')  #([x-axis],[y-axis],broken lines)
plt.plot(fpr,tpr,label='Decision Tree Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Decision Tree Classifier')
plt.show()
```

Here the accuracy score for decision tree classifier at random state 99 is 91%. Precision and recall for class 0 are 91% respectively which is equal. For class 1 91% and 90% respectively which gives good performance. When we check on AUC ROC curve, it is not in curve shape, it shows as triangular shape. 25% of data were given for testing.

Bernoulli NB



localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
jupyter credit_defaulter_model Last Checkpoint: 23 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
BernoulliNB Classifier
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
Type here to search
28°C Mostly sunny
ENG
01:00 PM
08-01-2022

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=99)
bnb.fit(x_train,y_train)
predbnb=bnb.predict(x_test)
acc_b=classification_report(y_test,predbnb)
print(acc_b)
print(confusion_matrix(y_test,predbnb))
```

```
              precision    recall  f1-score   support

           0       0.73      0.74      0.74     45979
           1       0.73      0.73      0.73     45737

    accuracy                           0.73     91716
   macro avg       0.73      0.73      0.73     91716
weighted avg       0.73      0.73      0.73     91716

[[33843 12136]
 [12261 33476]]
```

In [93]:

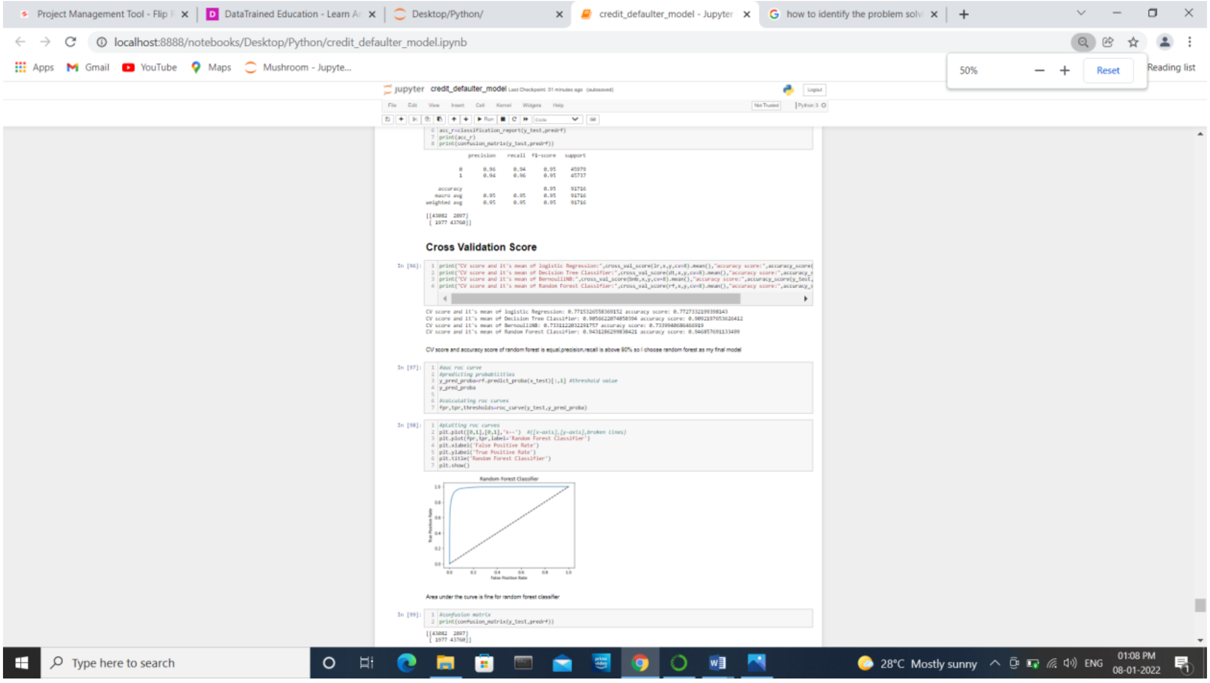
```
#auc roc curve
#predicting probabilities
y_pred_proba=bnb.predict_proba(x_test)[:,1] #threshold value
y_pred_proba

#calculating roc curves
fpr,tpr,thresholds=roc_curve(y_test,y_pred_proba)
```

In [94]:

```
#plotting roc curves
plt.plot([0,1],[0,1],'k--')  #([x-axis],[y-axis],broken lines)
plt.plot(fpr,tpr,label='BernoulliNB Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('BernoulliNB Classifier')
plt.show()
```

Here the accuracy score for model bernoulli naïve bayes is 73% which is low accuracy compared to other models. Precison and recall for class 0 is 73% and 74% respectivcely, for class 1 is 73% respectively which are equal. When we comes to AUC ROC curve which is not good at all. So we can conclude that bernoulli gives low accuracy compare to other model.

Random Forest Classifier


Project Management Tool - Flip
DataTrained Education - Learn A
Desktop/Python/
credit_defaulter_model - Jupyter
how to identify the problem solv
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
50%
Reset
Reading list
jupyter
credit_defaulter_model Last Checkpoint: 31 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Type here to search
28°C Mostly sunny
ENG
01:08 PM
08-01-2022

```
acc_r=classification_report(y_test,predrf)
print(acc_r)
print(confusion_matrix(y_test,predrf))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     45979
           1       0.94      0.96      0.95     45737

    accuracy                           0.95     91716
   macro avg       0.95      0.95      0.95     91716
weighted avg       0.95      0.95      0.95     91716

[[43082  2897]
 [ 1977 43760]]
```

Cross Validation Score

In [96]:

```
print("CV score and it's mean of logistic Regression:",cross_val_score(lr,x,y,cv=8).mean(),"accuracy score:",accuracy_score(
print("CV score and it's mean of Decision Tree Classifier:",cross_val_score(dt,x,y,cv=8).mean(),"accuracy score:",accuracy_
print("CV score and it's mean of BernoulliNB:",cross_val_score(bnb,x,y,cv=8).mean(),"accuracy score:",accuracy_score(y_test,
print("CV score and it's mean of Random Forest Classifier:",cross_val_score(rf,x,y,cv=8).mean(),"accuracy score:",accuracy_
```

```
CV score and it's mean of logistic Regression: 0.7715326558369152 accuracy score: 0.7727332199398143
CV score and it's mean of Decision Tree Classifier: 0.9056622074850394 accuracy score: 0.9092197653626412
CV score and it's mean of BernoulliNB: 0.7331122032291757 accuracy score: 0.7339940686466919
CV score and it's mean of Random Forest Classifier: 0.9431286299830421 accuracy score: 0.946857691133499
```

CV score and accuracy score of random forest is equal,precision,recall is above 90% so I choose random forest as my final model

In [97]:

```
#auc roc curve
#predicting probabilities
y_pred_proba=rf.predict_proba(x_test)[:,1] #threshold value
y_pred_proba

#calculating roc curves
fpr,tpr,thresholds=roc_curve(y_test,y_pred_proba)
```

In [98]:

```
#plotting roc curves
plt.plot([0,1],[0,1],'k--')  #([x-axis],[y-axis],broken lines)
plt.plot(fpr,tpr,label='Random Forest Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest Classifier')
plt.show()
```

Area under the curve is fine for random forest classifier

In [99]:

```
#confusion matrix
print(confusion_matrix(y_test,predrf))
```

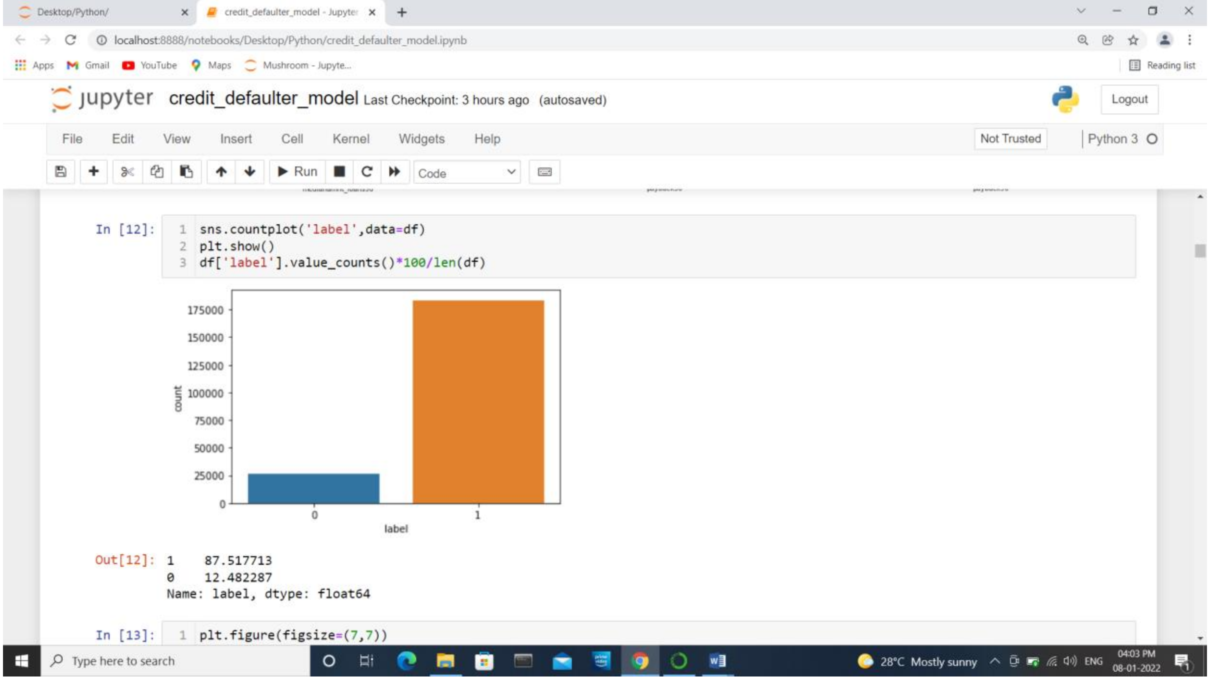
```
[[43082  2897]
 [ 1977 43760]]
```

Here we can observe the accuracy score of random forest is 95%, which is good among others. Precison and recall of random forest classifier for class 0 is 96% and 94% respectively, for class 1 is 94% and 96% respectively which we can conclude that good performance. When we comes to AUC ROC curve it is fine and good among comparing with other model. I choose random forest as my final model because it gives high accuracy score and cv score among other models.

- Key Metrics for success in solving problem under consideration

Accuracy score is used as one of the key metrics. Good accuracy score gives the high performance for prediction. We can evaluate the accuracy score of different model for better performance. Precision is also used for how much the model predicted the true positives. While recall is used for finding how many true positives for found by the model. The relation of precision and recall are used to find f1 score.

- Visualizations

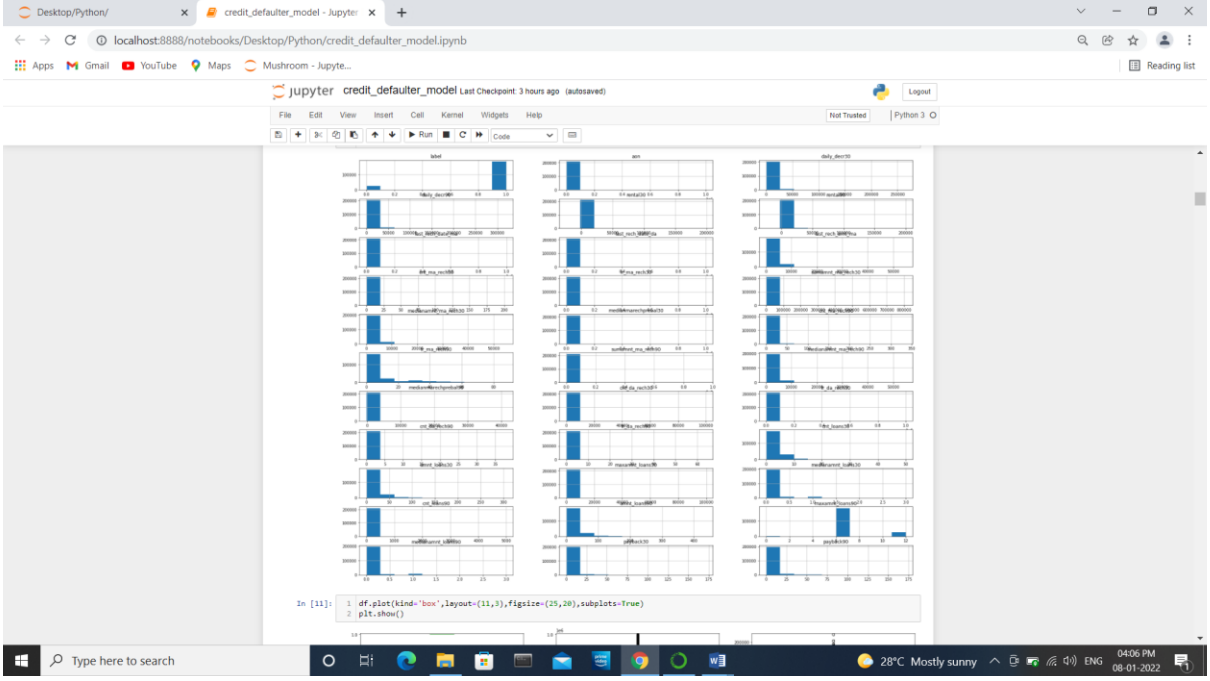


```
In [12]:  sns.countplot('label',data=df)
          plt.show()
          df['label'].value_counts()*100/len(df)
```

```
Out[12]: 1    87.517713
         0    12.482287
         Name: label, dtype: float64
```

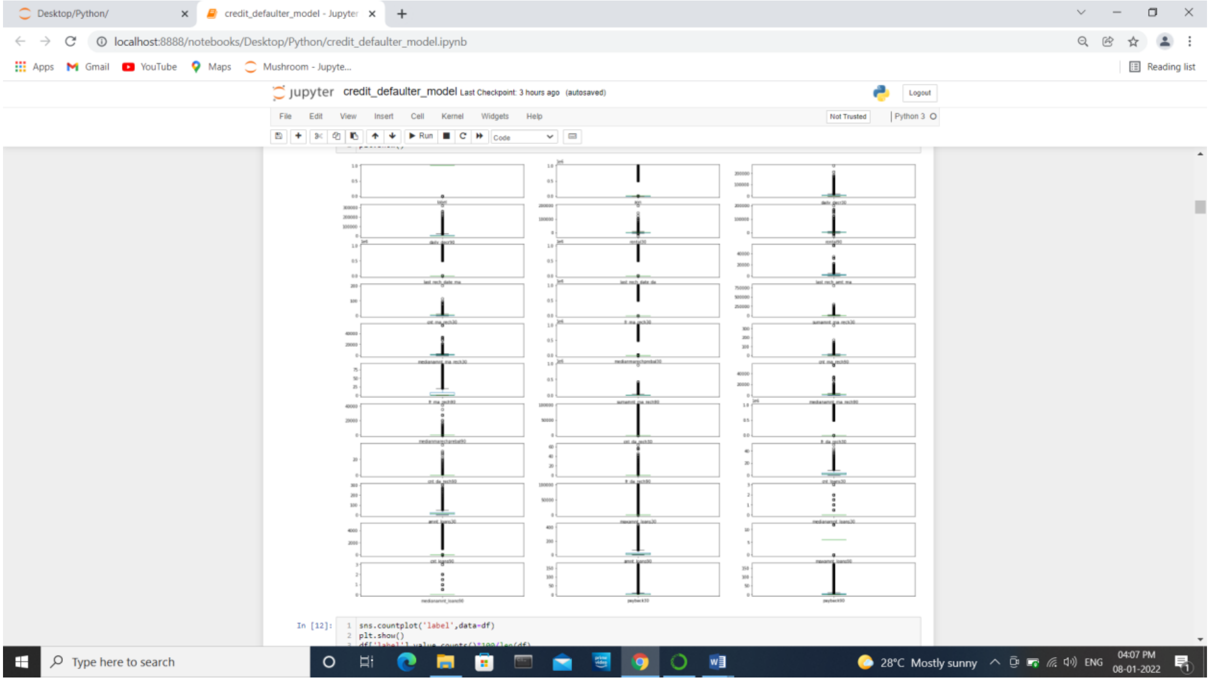
```
In [13]:  plt.figure(figsize=(7,7))
```

Here we check the value counts of the label using count plot. We can observe the imbalance of the class. Majority of the loans were paid back.



```
In [11]:  df.plot(kind='box',layout=(11,3),figsize=(25,20),subplots=True)
          plt.show()
```

Here histogram is used for checking Skewness of the features. Most of them were right skewed.

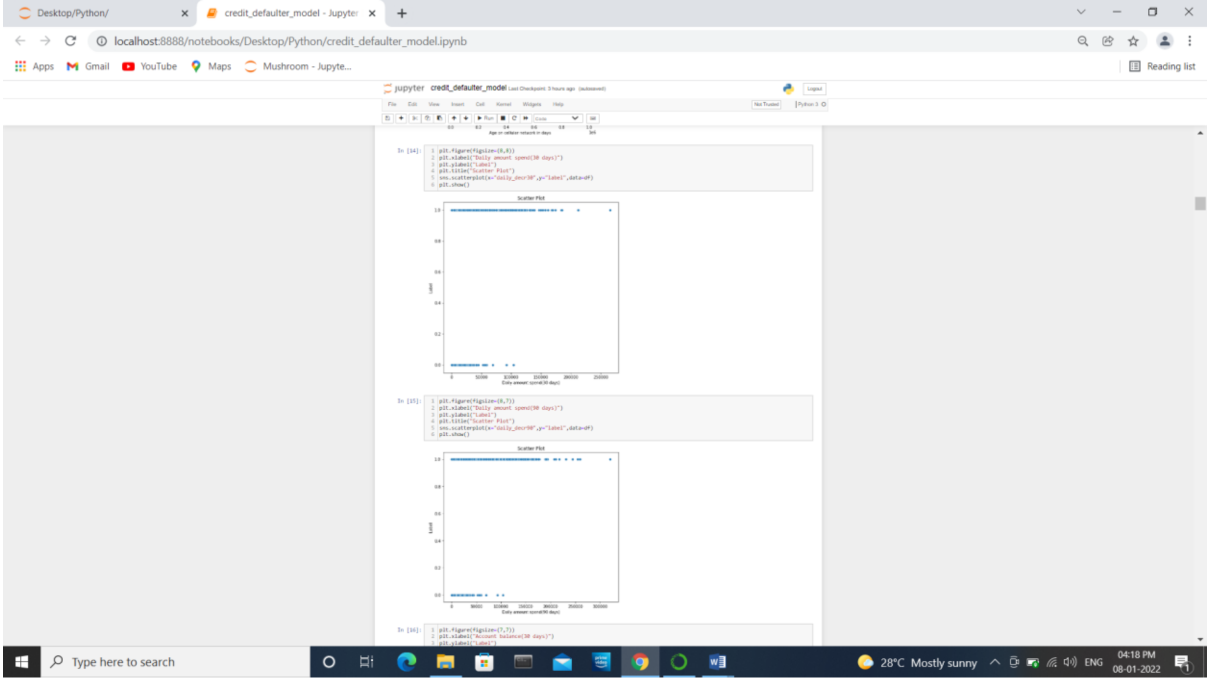


Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
jupyter credit_defaulter_model Last Checkpoint: 3 hours ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Logout
medianamnt_loans90
payback30
payback90

```
In [12]: sns.countplot('label',data=df)
         plt.show()
         df['label'].value_counts()*100/len(df)
```

Here boxplot is used for detecting outliers. We can observe the presence of outliers in dataset. If we try to remove outliers we have much loss of data, it may affect our prediction.

Now let’s compare the output and other features using scatter plot



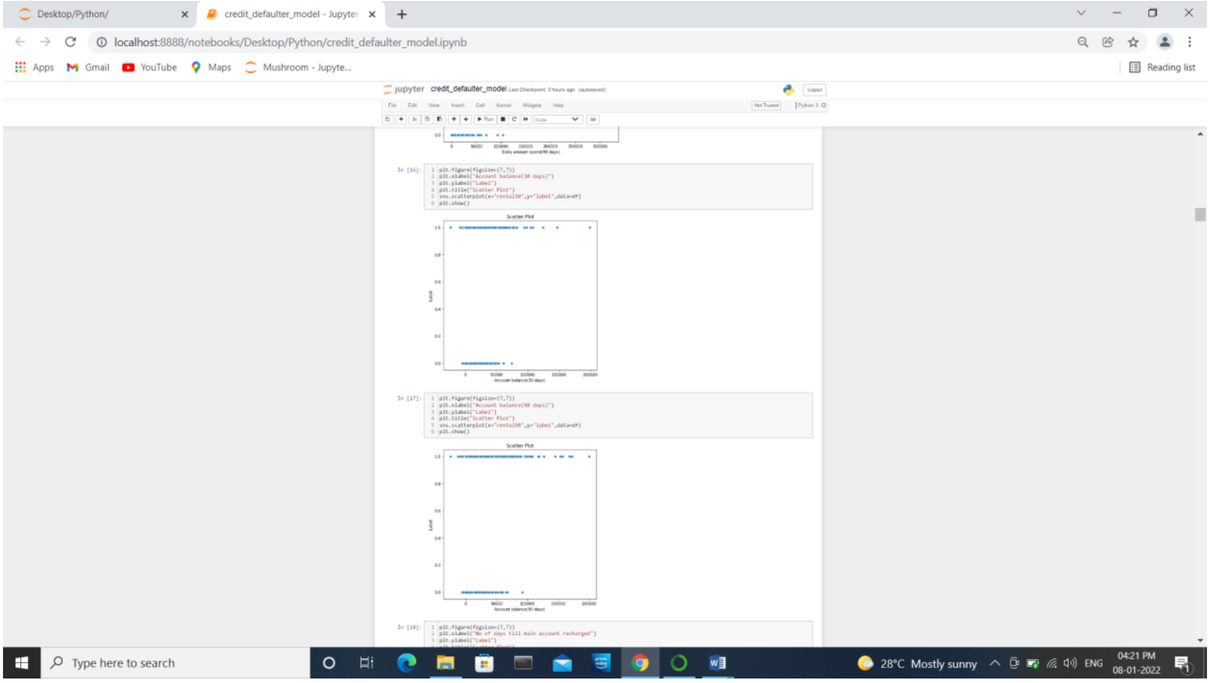
Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
jupyter credit_defaulter_model Last Checkpoint: 3 hours ago (autosaved)
Age on cellular network in days
Scatter Plot
Label
Daily amount spend(30 days)
Scatter Plot
Label
Daily amount spend(90 days)

```
In [14]: plt.figure(figsize=(8,8))
         plt.xlabel("Daily amount spend(30 days)")
         plt.ylabel("Label")
         plt.title("Scatter Plot")
         sns.scatterplot(x="daily_decr30",y="label",data=df)
         plt.show()
```

```
In [15]: plt.figure(figsize=(8,7))
         plt.xlabel("Daily amount spend(90 days)")
         plt.ylabel("Label")
         plt.title("Scatter Plot")
         sns.scatterplot(x="daily_decr90",y="label",data=df)
         plt.show()
```

```
In [16]: plt.figure(figsize=(7,7))
         plt.xlabel("Account balance(30 days)")
         plt.ylabel("Label")
```

Here we are comparing with daily amount spend by user in 30 days and daily amount spend by customer in 90 days. Here most of the users which spend more money have paid back the loan. So we can conclude that when the amount of spent increases the possibility of pay back of the loan increases.

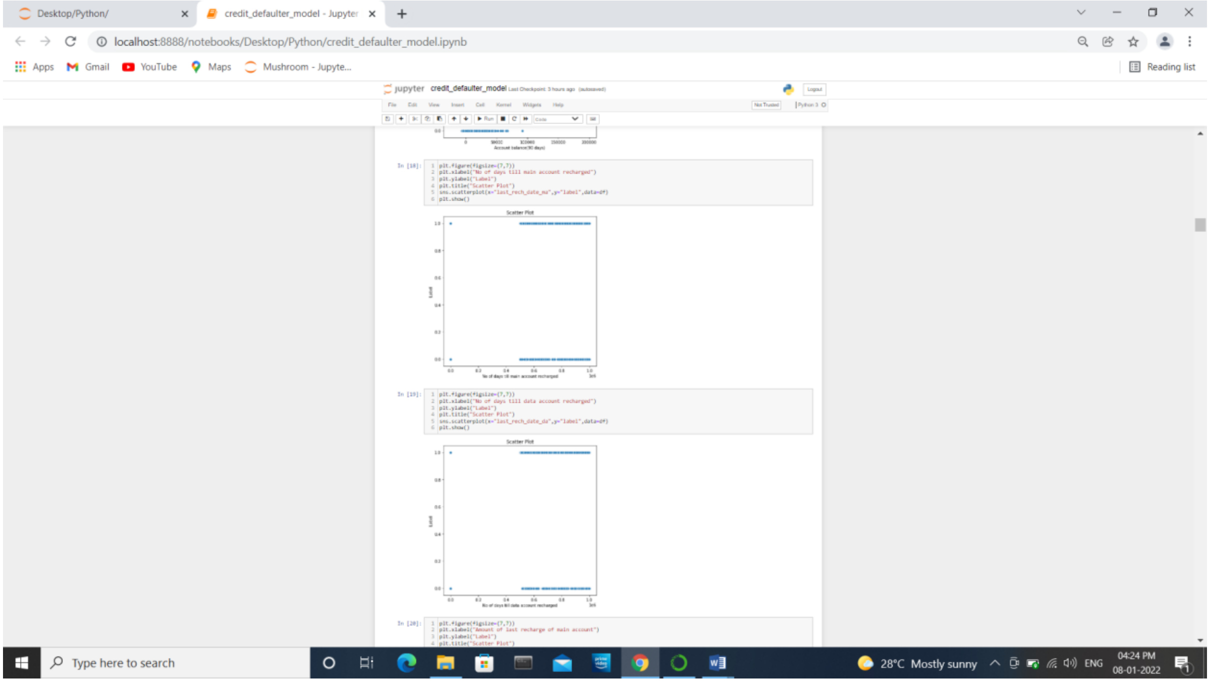


```
In [16]:
plt.figure(figsize=(7,7))
plt.xlabel("Account balance(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="rental30",y="label",data=df)
plt.show()
```

```
In [17]:
plt.figure(figsize=(7,7))
plt.xlabel("Account balance(90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="rental90",y="label",data=df)
plt.show()
```

```
In [18]:
plt.figure(figsize=(7,7))
plt.xlabel("No of days till main account recharged")
plt.ylabel("Label")
```

Here we compared output with balance of the account in last 30 days and 90 days. The customer who have more balance in account were paid back. Others have less possibility to pay the loan back.

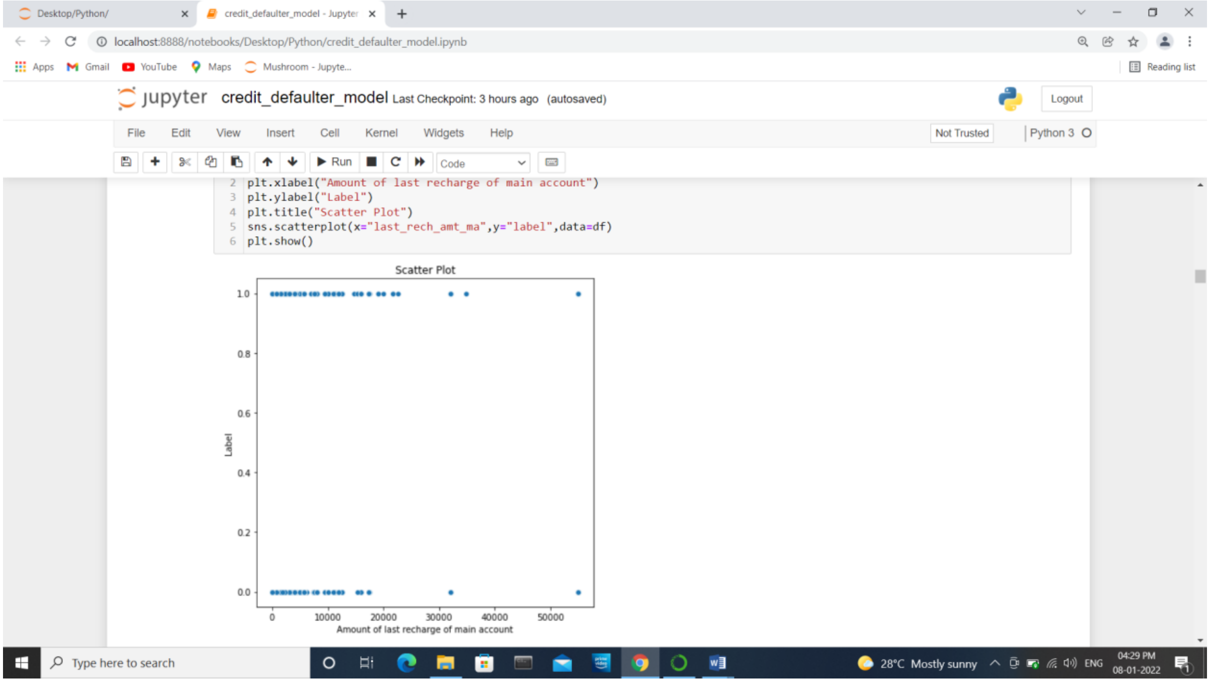


```
In [18]:
plt.figure(figsize=(7,7))
plt.xlabel("No of days till main account recharged")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="last_rech_date_ma",y="label",data=df)
plt.show()
```

```
In [19]:
plt.figure(figsize=(7,7))
plt.xlabel("No of days till data account recharged")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="last_rech_date_da",y="label",data=df)
plt.show()
```

```
In [20]:
plt.figure(figsize=(7,7))
plt.xlabel("Amount of last recharge of main account")
plt.ylabel("Label")
plt.title("Scatter Plot")
```

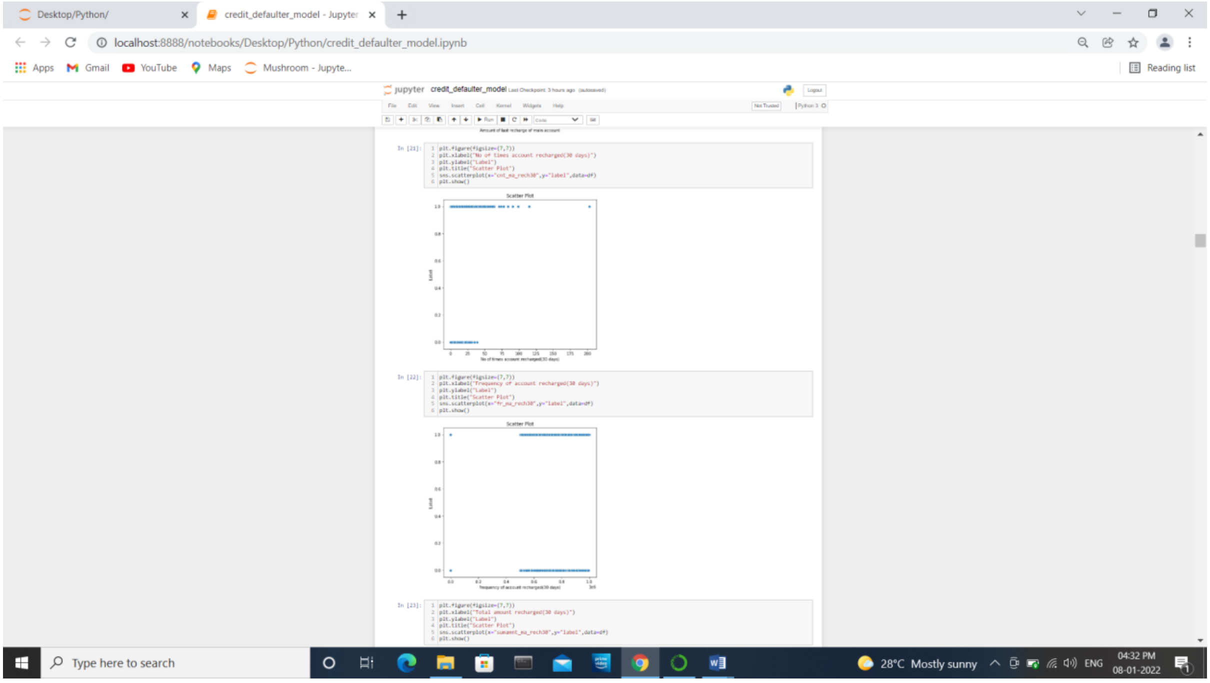
Here we are comparing the how many days before the main and data account were recharged. Here these features equally affect the output. That is the customer who pay back the loan or the customer who does not pay back the loan have same number of days get recharge to main and data account.



Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter
credit_defaulter_model Last Checkpoint: 3 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code

```
plt.xlabel("Amount of last recharge of main account")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="last_rech_amt_ma",y="label",data=df)
plt.show()
```

Now we can compare the label with the feature amount which is recharged last. The customer who pay back the loan have recharged large amount to main account.

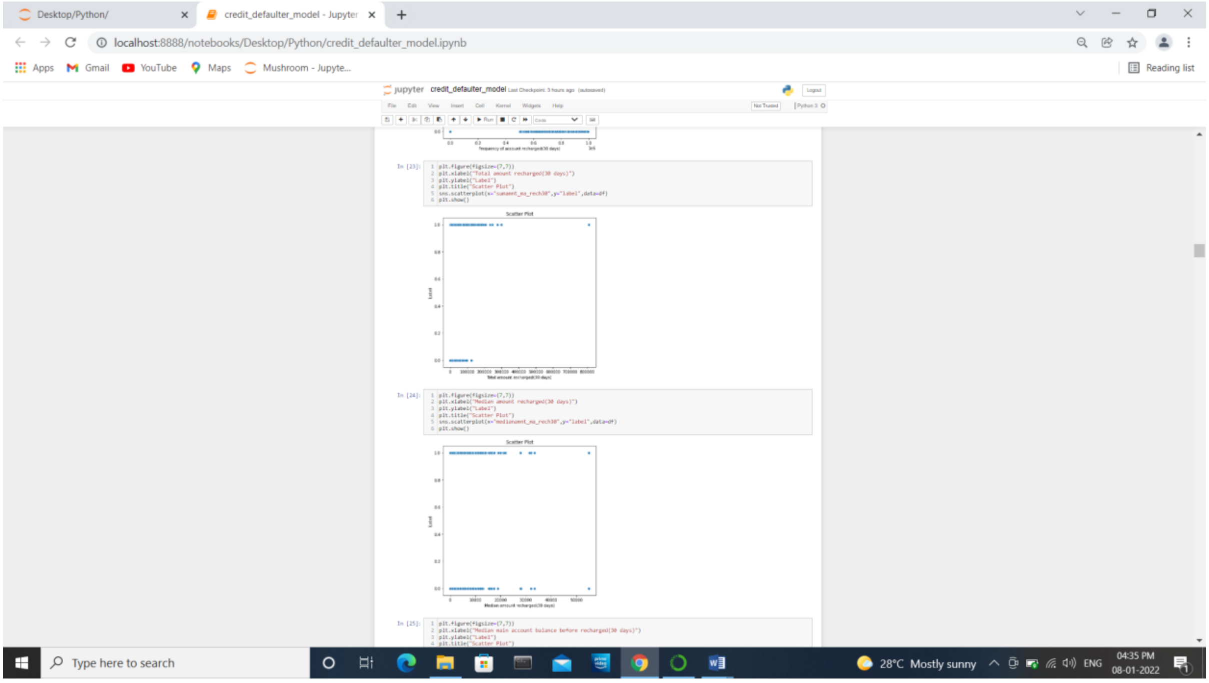


```
In [21]:
plt.figure(figsize=(7,7))
plt.xlabel("No of times account recharged(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="cnt_ma_rech30",y="label",data=df)
plt.show()
```

```
In [22]:
plt.figure(figsize=(7,7))
plt.xlabel("Frequency of account recharged(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="fr_ma_rech30",y="label",data=df)
plt.show()
```

```
In [23]:
plt.figure(figsize=(7,7))
plt.xlabel("Total amount recharged(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="sumamnt_ma_rech30",y="label",data=df)
plt.show()
```

Now we can compare number of times account recharged and frequency of account recharged in last 30 days. The customer who pay back the loan were recharged most number of times in last 30 days. While the frequency of account recharged were equal for the customers who pay back the loan who did not pay back.

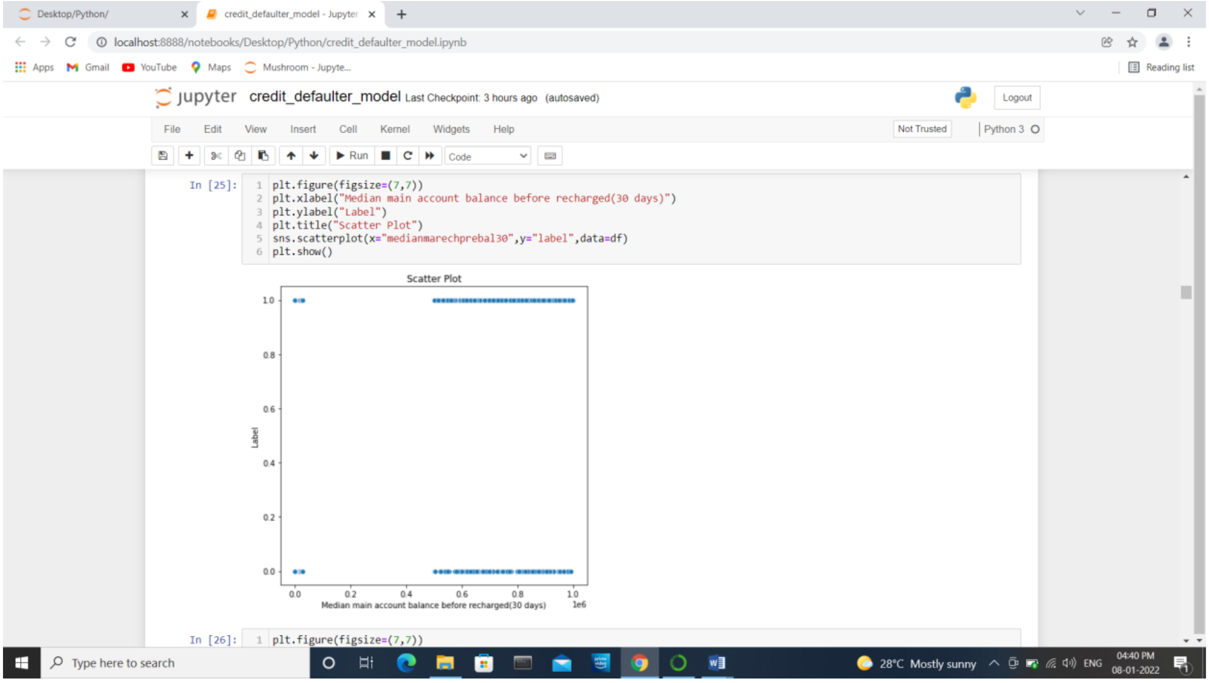


```
In [23]:
plt.figure(figsize=(7,7))
plt.xlabel("Total amount recharged(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="sumamnt_ma_rech30",y="label",data=df)
plt.show()
```

```
In [24]:
plt.figure(figsize=(7,7))
plt.xlabel("Median amount recharged(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="medianamnt_ma_rech30",y="label",data=df)
plt.show()
```

```
In [25]:
plt.figure(figsize=(7,7))
plt.xlabel("Median main account balance before recharged(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
```

Here we can compare the features total amount of recharge and median amount of recharge done in last 30 days. The customer who have pay back the loan have high amount of recharges done in 30 days while the median amount of recharges are equal for both customers who pay back and who didn’t.



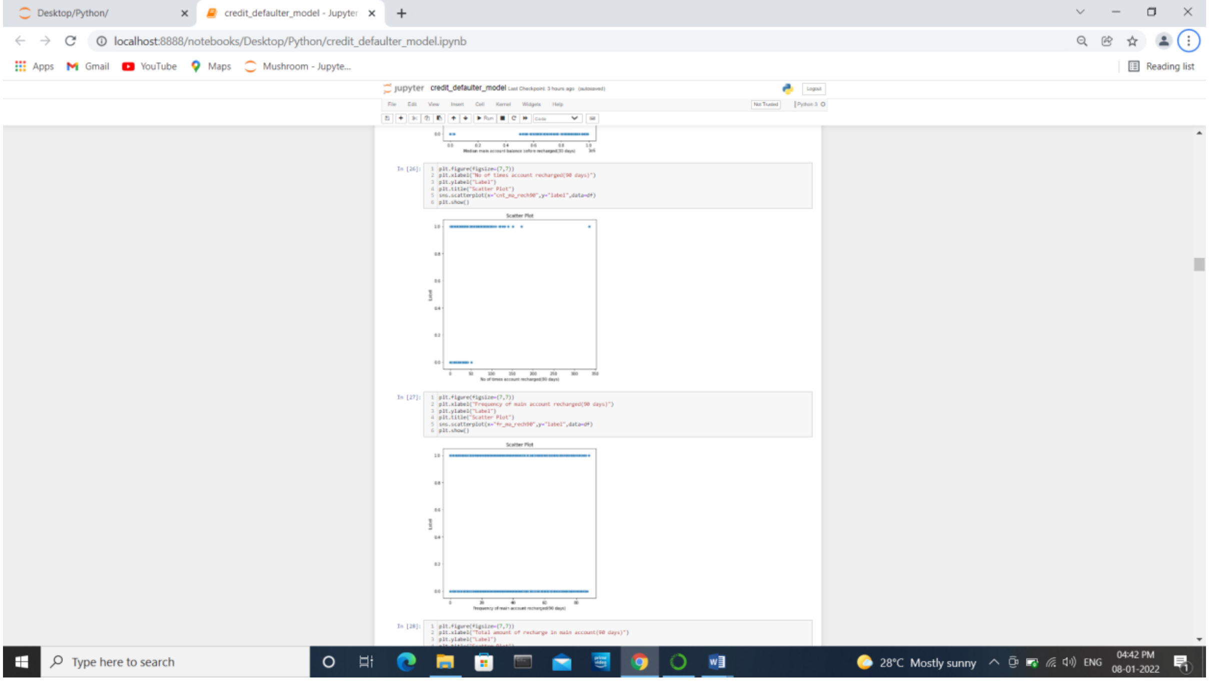
In [25]:

```
plt.figure(figsize=(7,7))
plt.xlabel("Median main account balance before recharged(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="medianmarechprebal30",y="label",data=df)
plt.show()
```

In [26]:

```
plt.figure(figsize=(7,7))
```

Here we are comparing the feature median of main account balance before recharging in 30 days. We can observe the median of account balance of both customer who pay back the loan and who doesn’t.



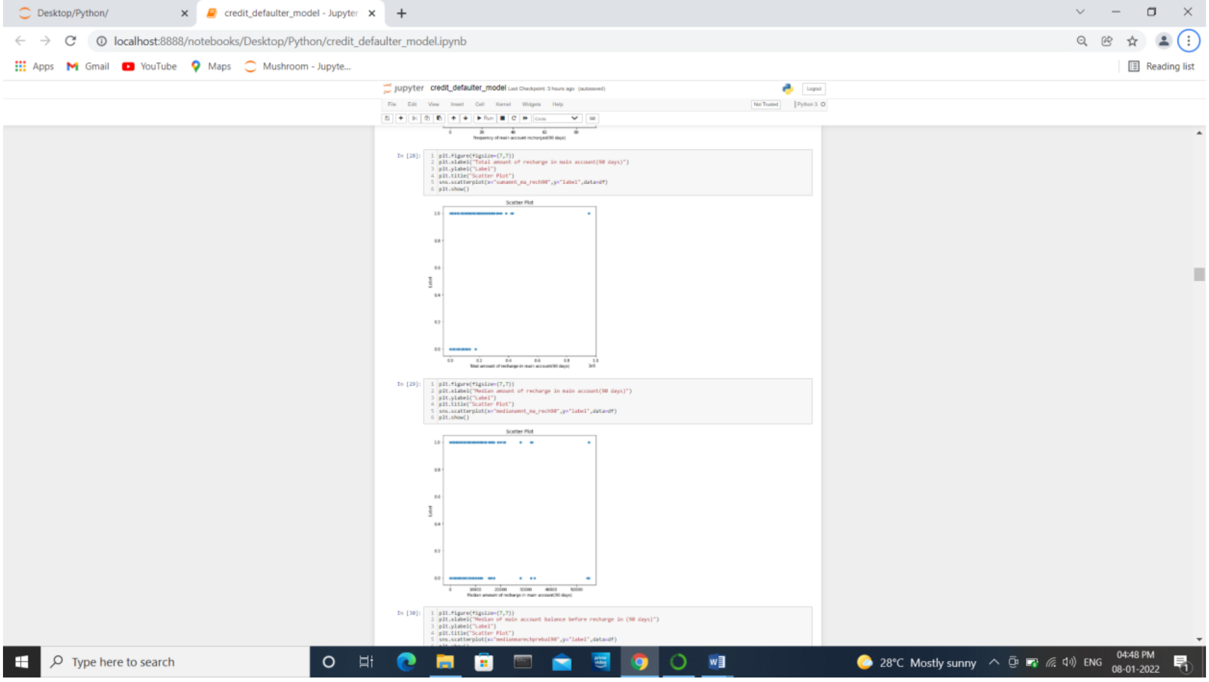
Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter
credit_defaulter_model Last Checkpoint: 3 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Median main account balance before recharged(30 days)
Scatter Plot
Label
No of times account recharged(90 days)
Frequency of main account recharged(90 days)
Type here to search
28°C Mostly sunny
ENG
04:42 PM
08-01-2022

In [26]:
```
plt.figure(figsize=(7,7))
plt.xlabel("No of times account recharged(90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="cnt_ma_rech90",y="label",data=df)
plt.show()
```

In [27]:
```
plt.figure(figsize=(7,7))
plt.xlabel("Frequency of main account recharged(90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="fr_ma_rech90",y="label",data=df)
plt.show()
```

In [28]:
```
plt.figure(figsize=(7,7))
plt.xlabel("Total amount of recharge in main account(90 days)")
plt.ylabel("Label")
```

Here we are comparing the features number of times account recharged and frequency of main account recharged in last 90 days. We can observe that the customer who pay back the loan were the most number of times account were recharged in last 90 days. And frequency of main account recharged were equal for both customer who pay back the loan and who didn’t.



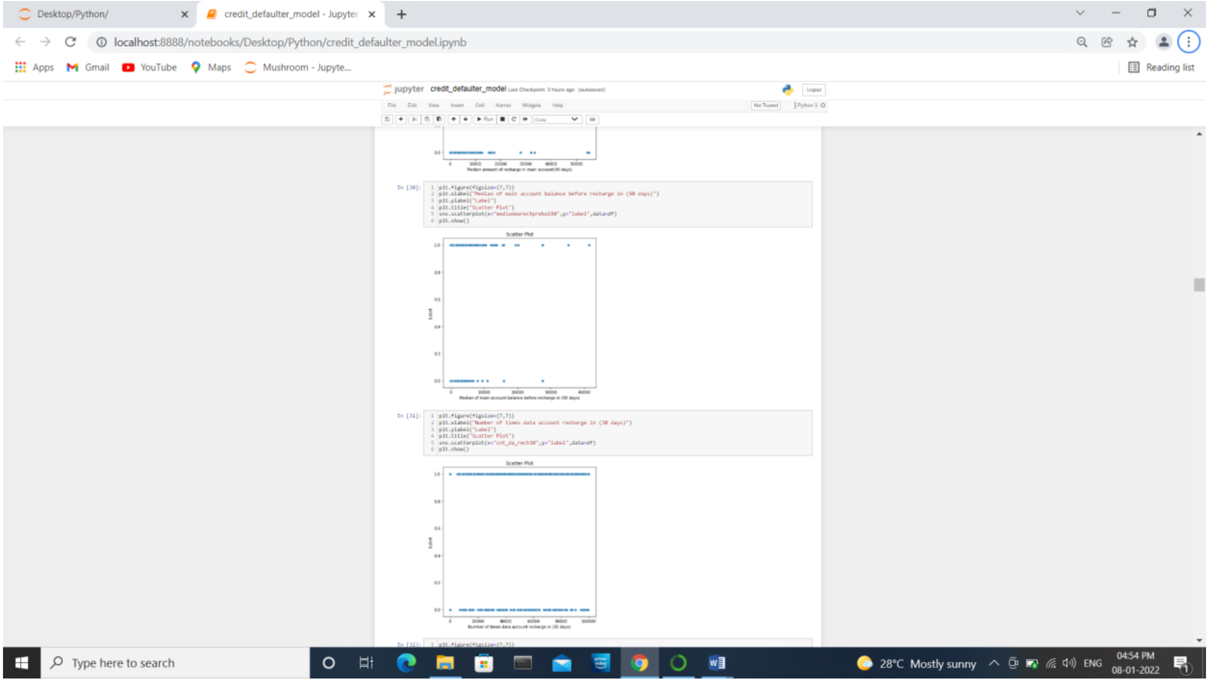
Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter
credit_defaulter_model Last Checkpoint: 3 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Frequency of main account recharged(90 days)
Scatter Plot
Label
Total amount of recharge in main account(90 days)
Median amount of recharge in main account(90 days)
Type here to search
28°C Mostly sunny
ENG
04:48 PM
08-01-2022

In [28]:
```
plt.figure(figsize=(7,7))
plt.xlabel("Total amount of recharge in main account(90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="sumamnt_ma_rech90",y="label",data=df)
plt.show()
```

In [29]:
```
plt.figure(figsize=(7,7))
plt.xlabel("Median amount of recharge in main account(90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="medianamnt_ma_rech90",y="label",data=df)
plt.show()
```

In [30]:
```
plt.figure(figsize=(7,7))
plt.xlabel("Median of main account balance before recharge in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="medianmarechprebal90",y="label",data=df)
```

Here we can compare the features total amount of recharge done in last 90 days and median amount of recharge in last 90 days. We can observe that those who pay back the loan have high amount of recharge done in last 90 days and also the median amount of recharge is high in customer's account who pay back the loan.

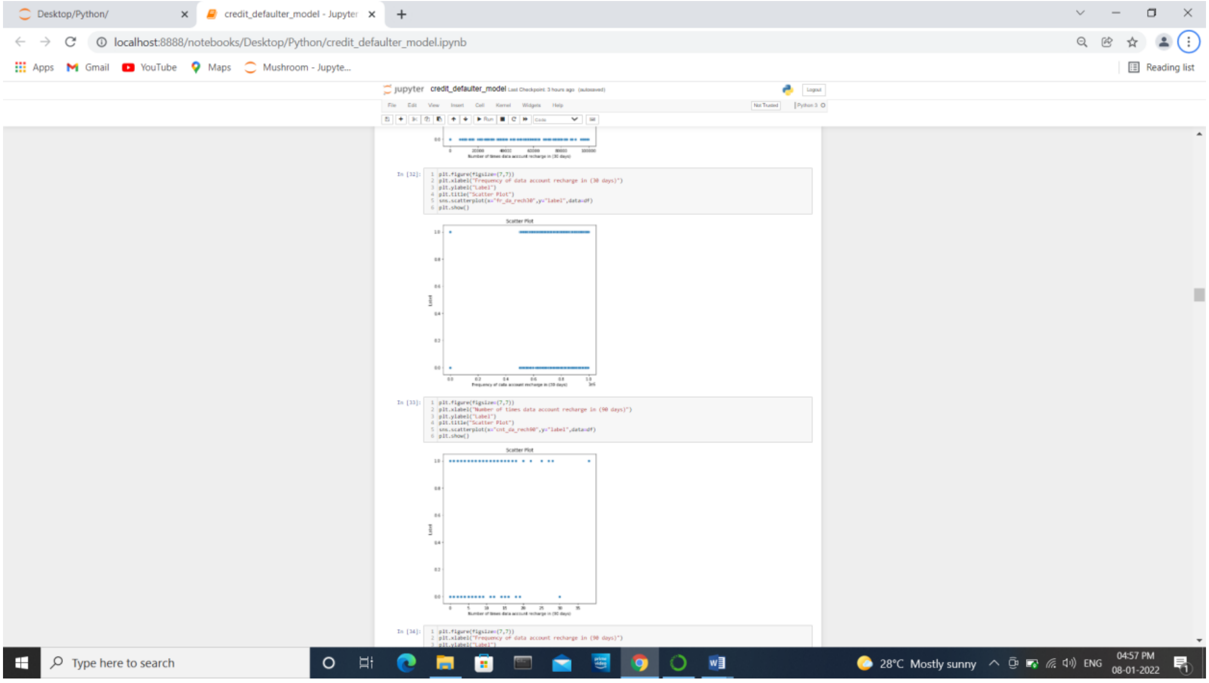


```
In [30]:
plt.figure(figsize=(7,7))
plt.xlabel("Median of main account balance before recharge in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="medianmarechprebal90",y="label",data=df)
plt.show()
```

```
In [31]:
plt.figure(figsize=(7,7))
plt.xlabel("Number of times data account recharge in (30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="cnt_da_rech30",y="label",data=df)
plt.show()
```

Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter credit_defaulter_model Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
0.0
0 10000 20000 30000 40000 50000
Median amount of recharge in main account(90 days)
Scatter Plot
1.0
0.8
0.6
0.4
0.2
0.0
Label
0 10000 20000 30000 40000
Median of main account balance before recharge in (90 days)
Scatter Plot
1.0
0.8
0.6
0.4
0.2
0.0
Label
0 20000 40000 60000 80000 100000
Number of times data account recharge in (30 days)
In [32]: plt.figure(figsize=(7,7))
Type here to search
28°C Mostly sunny
ENG
04:54 PM
08-01-2022

Here we are comparing the features median of main account before recharge in 90 days and number of times data account recharged last 30 days. We can observe that median of main account before recharge in 30 days were high in the account of customer's who pay back the loan. While the number of times data account recharge is equal for the customer who pay back the loan and who didn't.



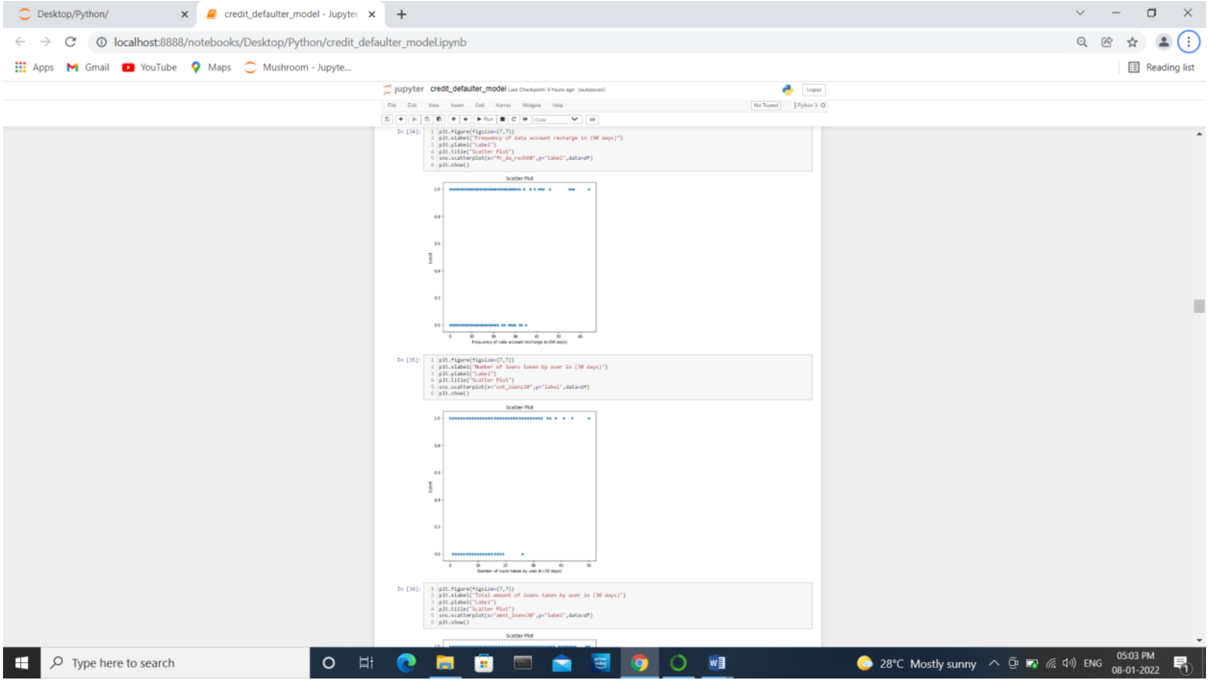
Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter credit_defaulter_model Last Checkpoint: 3 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Number of times data account recharge in (30 days)
Scatter Plot
Label
Frequency of data account recharge in (30 days)
Scatter Plot
Label
Number of times data account recharge in (90 days)
Type here to search
28°C Mostly sunny
ENG
04:57 PM
08-01-2022

In [32]:
```python
plt.figure(figsize=(7,7))
plt.xlabel("Frequency of data account recharge in (30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="fr_da_rech30",y="label",data=df)
plt.show()
```

In [33]:
```python
plt.figure(figsize=(7,7))
plt.xlabel("Number of times data account recharge in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="cnt_da_rech90",y="label",data=df)
plt.show()
```

In [34]:
```python
plt.figure(figsize=(7,7))
plt.xlabel("Frequency of data account recharge in (90 days)")
plt.ylabel("Label")
```

Here we can compare the features frequency of data account recharged in last 30 days and number of times data account recharged in 90 days. We can observe that frequency of data account recharged were equal to the customer those who pay back the loan and who didn’t. Up to 20 times were data account recharged by customer who pay back the loan.



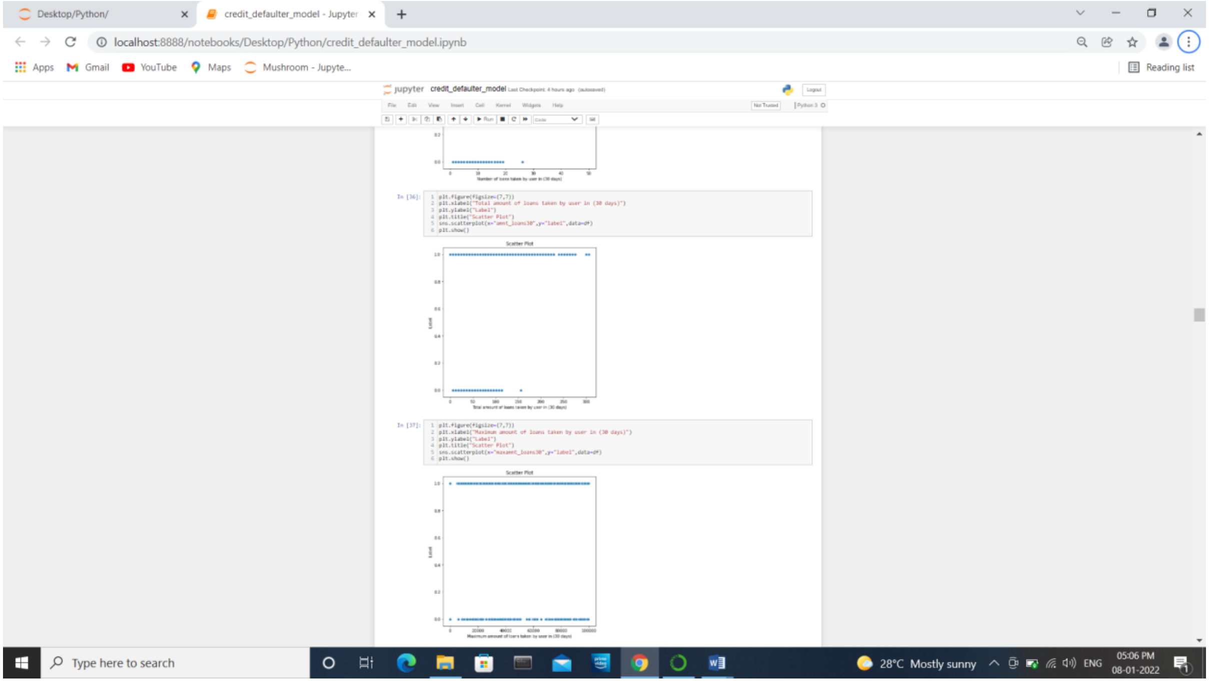
Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter credit_defaulter_model Last Checkpoint: 4 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Scatter Plot
Label
Frequency of data account recharge in (90 days)
Scatter Plot
Label
Number of loans taken by user in (30 days)
Scatter Plot
Type here to search
28°C Mostly sunny
ENG
05:03 PM
08-01-2022

In [34]:
```python
plt.figure(figsize=(7,7))
plt.xlabel("Frequency of data account recharge in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="fr_da_rech90",y="label",data=df)
plt.show()
```

In [35]:
```python
plt.figure(figsize=(7,7))
plt.xlabel("Number of loans taken by user in (30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="cnt_loans30",y="label",data=df)
plt.show()
```

In [36]:
```python
plt.figure(figsize=(7,7))
plt.xlabel("Total amount of loans taken by user in (30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="amnt_loans30",y="label",data=df)
plt.show()
```

Here we can compare the features frequency of data account recharge in last 90 days and number of loans taken by the user in last 30 days. The customer who pay back the loan have the high frequency of data account recharged. The customer who pay back the loan have taken highest number of loans in last 30 days.

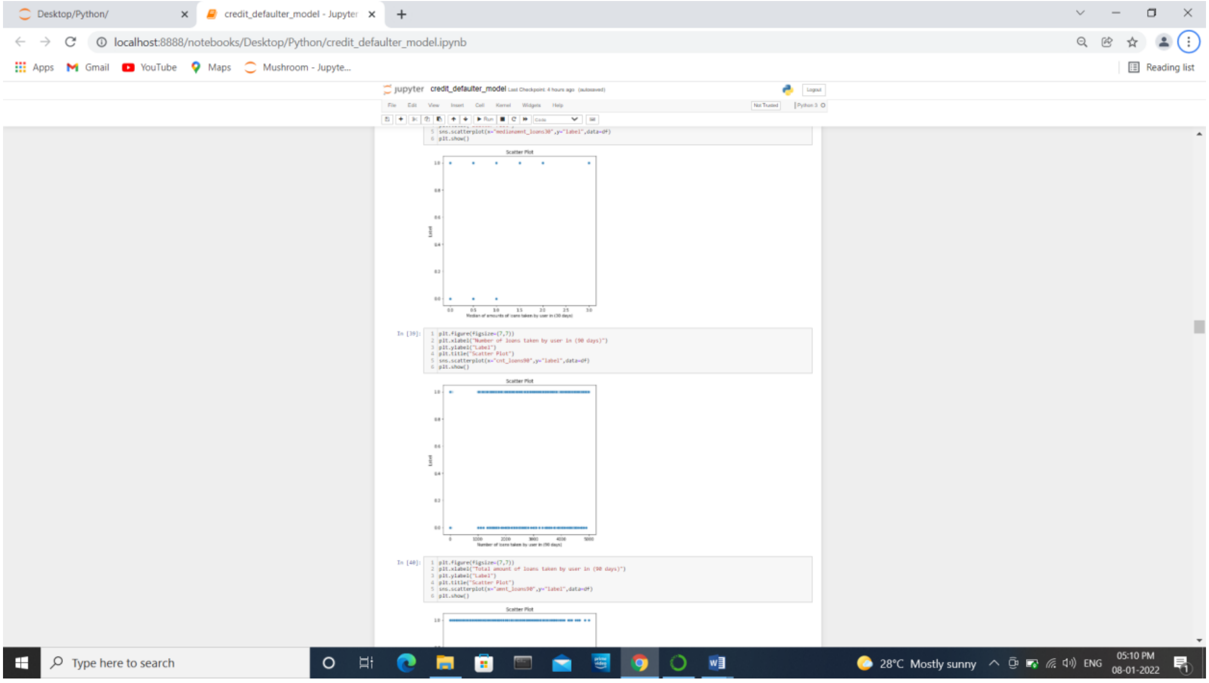


Desktop/Python/
credit_defaulter_model - Jupyter
localhost:8888/notebooks/Desktop/Python/credit_defaulter_model.ipynb
Apps
Gmail
YouTube
Maps
Mushroom - Jupyte...
Reading list
jupyter
credit_defaulter_model
Last Checkpoint: 4 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Number of loans taken by user in (30 days)
Scatter Plot
Label
Total amount of loans taken by user in (30 days)
Scatter Plot
Label
Maximum amount of loans taken by user in (30 days)
Type here to search
28°C Mostly sunny
ENG
05:06 PM
08-01-2022

```
In [36]:  plt.figure(figsize=(7,7))
          plt.xlabel("Total amount of loans taken by user in (30 days)")
          plt.ylabel("Label")
          plt.title("Scatter Plot")
          sns.scatterplot(x="amnt_loans30",y="label",data=df)
          plt.show()
```

```
In [37]:  plt.figure(figsize=(7,7))
          plt.xlabel("Maximum amount of loans taken by user in (30 days)")
          plt.ylabel("Label")
          plt.title("Scatter Plot")
          sns.scatterplot(x="maxamnt_loans30",y="label",data=df)
          plt.show()
```

Now we can compare the features total amount of loans taken by user and maximum amount of loans taken by the user in last 30 days with the output. We can observe that the customer who pay back the loan have the highest total amount of loans. While the maximum amount of loan taken by the user are equal in both who pay back the loan and who didn’t.



```
sns.scatterplot(x="medianamnt_loans30",y="label",data=df)
plt.show()
```

Scatter Plot
Label
Median of amounts of loans taken by user in (30 days)

In [39]:

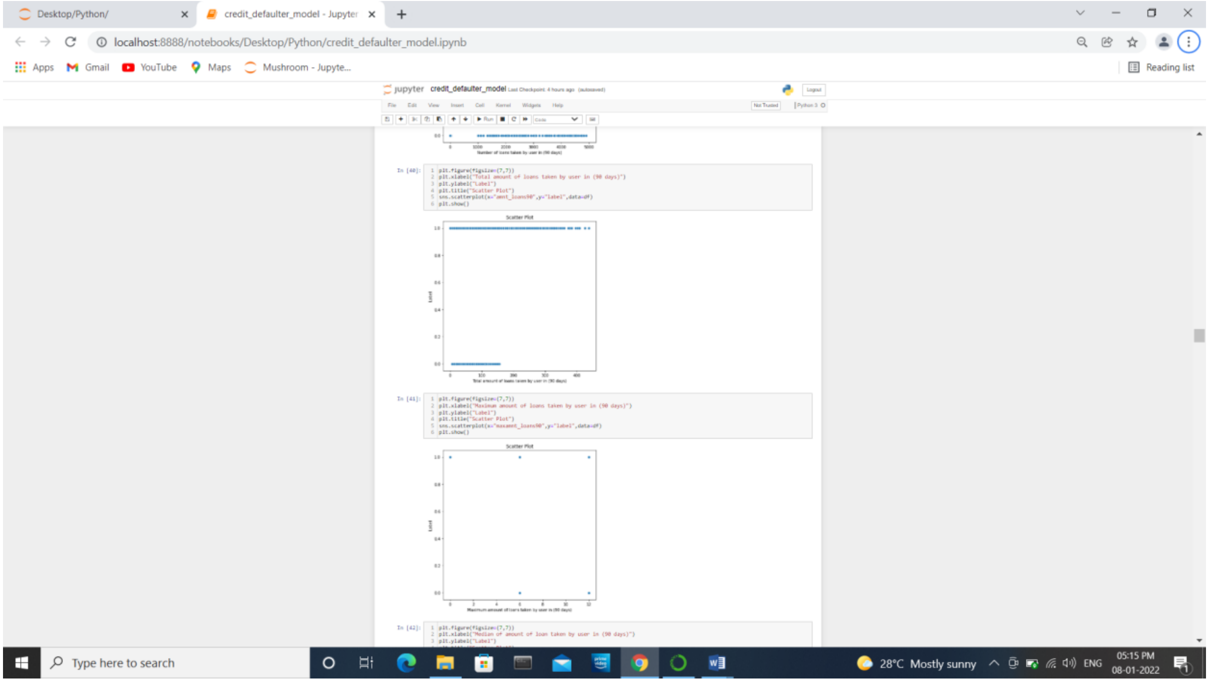
```
plt.figure(figsize=(7,7))
plt.xlabel("Number of loans taken by user in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="cnt_loans90",y="label",data=df)
plt.show()
```

Scatter Plot
Label
Number of loans taken by user in (90 days)

In [40]:

```
plt.figure(figsize=(7,7))
plt.xlabel("Total amount of loans taken by user in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="amnt_loans90",y="label",data=df)
plt.show()
```

Now we can compare the features median amount of loans taken by user in last 30 days and number of loans taken by the user in last 90days. We can observe that the user who have pay back the loan have high median of amount loans and number of loans taken by the user were equal to the both who pay back the loan and who didn't.



In [40]:

```
plt.figure(figsize=(7,7))
plt.xlabel("Total amount of loans taken by user in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="amnt_loans90",y="label",data=df)
plt.show()
```

Scatter Plot
Label
Total amount of loans taken by user in (90 days)

In [41]:

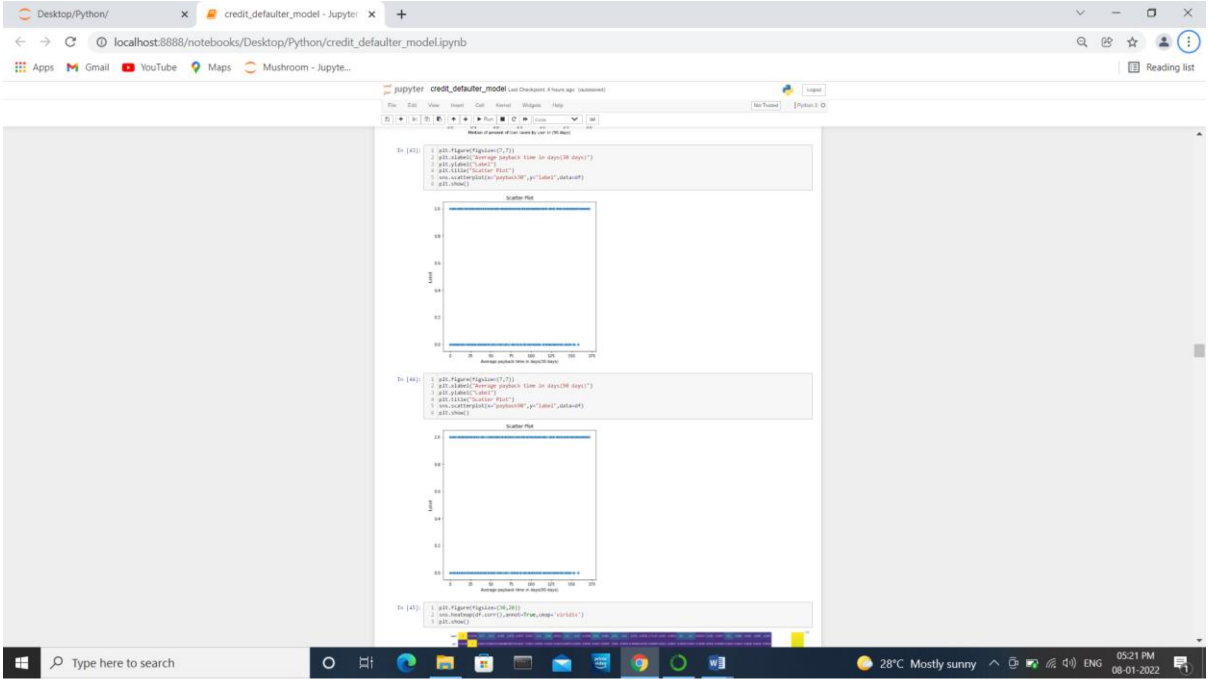
```
plt.figure(figsize=(7,7))
plt.xlabel("Maximum amount of loans taken by user in (90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="maxamnt_loans90",y="label",data=df)
plt.show()
```

Scatter Plot
Label
Maximum amount of loans taken by user in (90 days)

In [42]:

```
plt.figure(figsize=(7,7))
plt.xlabel("Median of amount of loan taken by user in (90 days)")
plt.ylabel("Label")
```

Now we can compare the features total amount of loan taken by the user and maximum amount loans taken by the user in last 90 days. The user who pay back the loan have highest total of amount of loan and maximum amount of loans is equal for both who paid and who didn't.

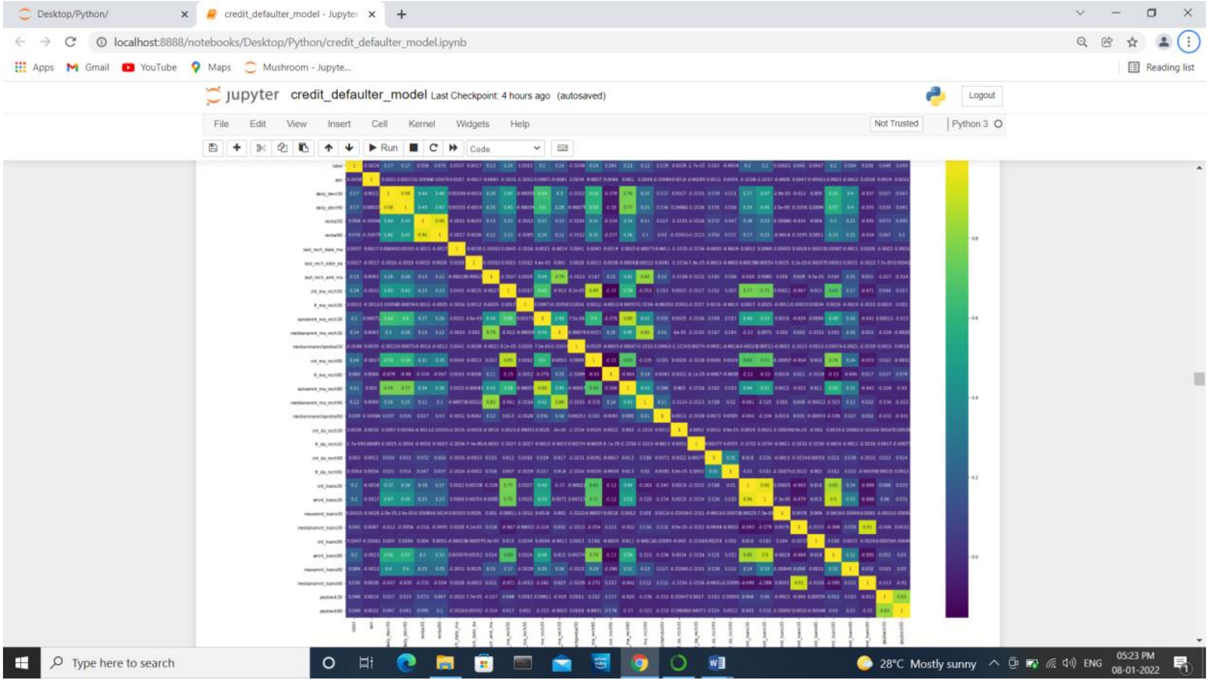


```
In [43]:
plt.figure(figsize=(7,7))
plt.xlabel("Average payback time in days(30 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="payback30",y="label",data=df)
plt.show()
```

```
In [44]:
plt.figure(figsize=(7,7))
plt.xlabel("Average payback time in days(90 days)")
plt.ylabel("Label")
plt.title("Scatter Plot")
sns.scatterplot(x="payback90",y="label",data=df)
plt.show()
```

```
In [45]:
plt.figure(figsize=(30,20))
sns.heatmap(df.corr(),annot=True,cmap='viridis')
plt.show()
```

Now we can compare the average payback time in last 30 days and 90 days. The user who have pay back the loan have high average payback in both 30 days and 90 days.



Now we can check the correlation, the above visualization is Heatmap. The green colour shows the features how correlated to other features. The green colour shows how highly correlated to other. Here the 30 days features and 90 days features were highly correlated to each other.

- Interpretation of the Results

From above visualizations we can conclude that the user who have balance in the main account before taking loan will pay back the loan within 5 days. And also who have recharged the main account and data account have the possibility to pay back the account.

CONCLUSION

- Key Findings and Conclusions of the Study

The user have balance in main account and how many times the account got recharged and the time for pay backing the loan. Company want to concentrate the on these features of the user. Whether the user have the capability to pay back the loan. What is the balance of main account of the user, average time to take the user to pay back if the user pay back the loan within 5 days he is non-defaulter. Consistent recharge of account gives the pay back of the loan.

- Learning Outcomes of the Study in respect of Data Science

Visualizations gives a better understanding of data. How the features were related to each other and how other features related to the output. Data cleaning gives better accuracy, how well the data is cleaned gives better accuracy. Using various algorithms, gives a better understanding which gives better accuracy and which algorithm will perform well. We can compare the accuracy of each algorithm and which gives high accuracy and we can select that algorithm as our final model. While working the algorithms with this project random forest gives the result little bit Slow but it gives better accuracy than other model.

- Limitations of this work and Scope for Future Work

As this micro credit is for low income families, every family is not capable for paying back the loan. The amount of loan is low, but the families will not have the capability of pay back the loan. The families are living with daily wages when they loss their job or not go for work one day. It may affect their incomes. To overcome this try to extend the payback days it will give a better result.