**Loan Application Status Prediction**

**Submitted by: Vinayak Ratan**

# **Problem Definition**

This dataset includes details of applicants who have applied for loan. The dataset includes details like credit history, loan amount, their income, dependents etc.

Independent Variables:

* Loan\_ID – unique loan ID for each applicant
* Gender – male/female
* Married – applicant marital status (Yes/No)
* Dependents – dependents of the applicant
* Education – applicant graduated or not graduated
* Self\_Employed – self-employed (Yes/No)
* ApplicantIncome – income of an applicant
* CoapplicantIncome – income of the co-applicant
* Loan\_Amount – loan amount of an applicant in thousands
* Loan\_Amount\_Term – loan amount term in months
* Credit History – credit history of an applicant
* Property\_Area – area Semi-urban/urban/rural

Dependent Variable

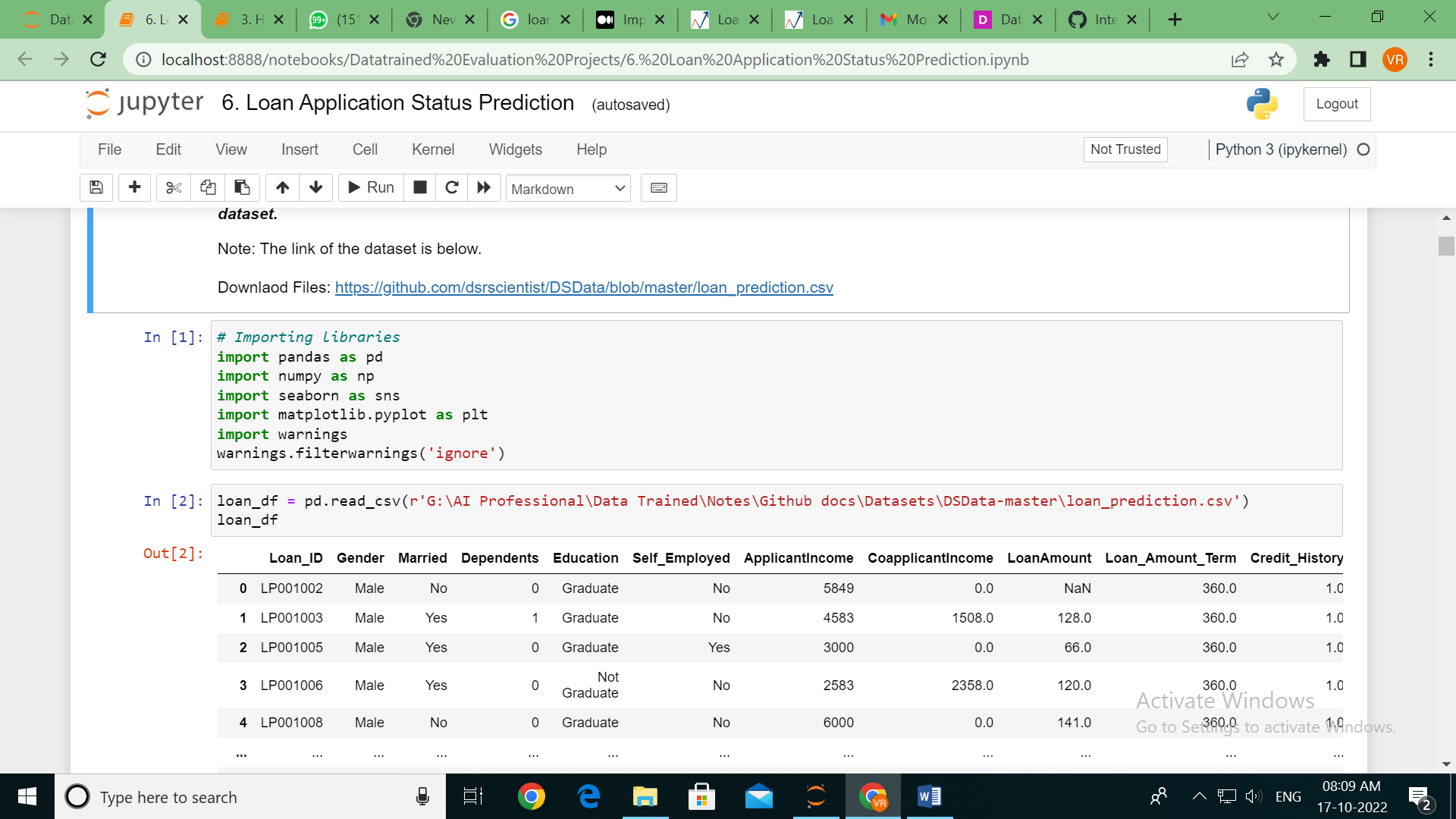
* Loan\_Status – loan status of an applicant (Y – approved, N – not approved)

You have to build a model that can predict whether the loan of the applicant will be approved or not on the basis of the details provided in the dataset.

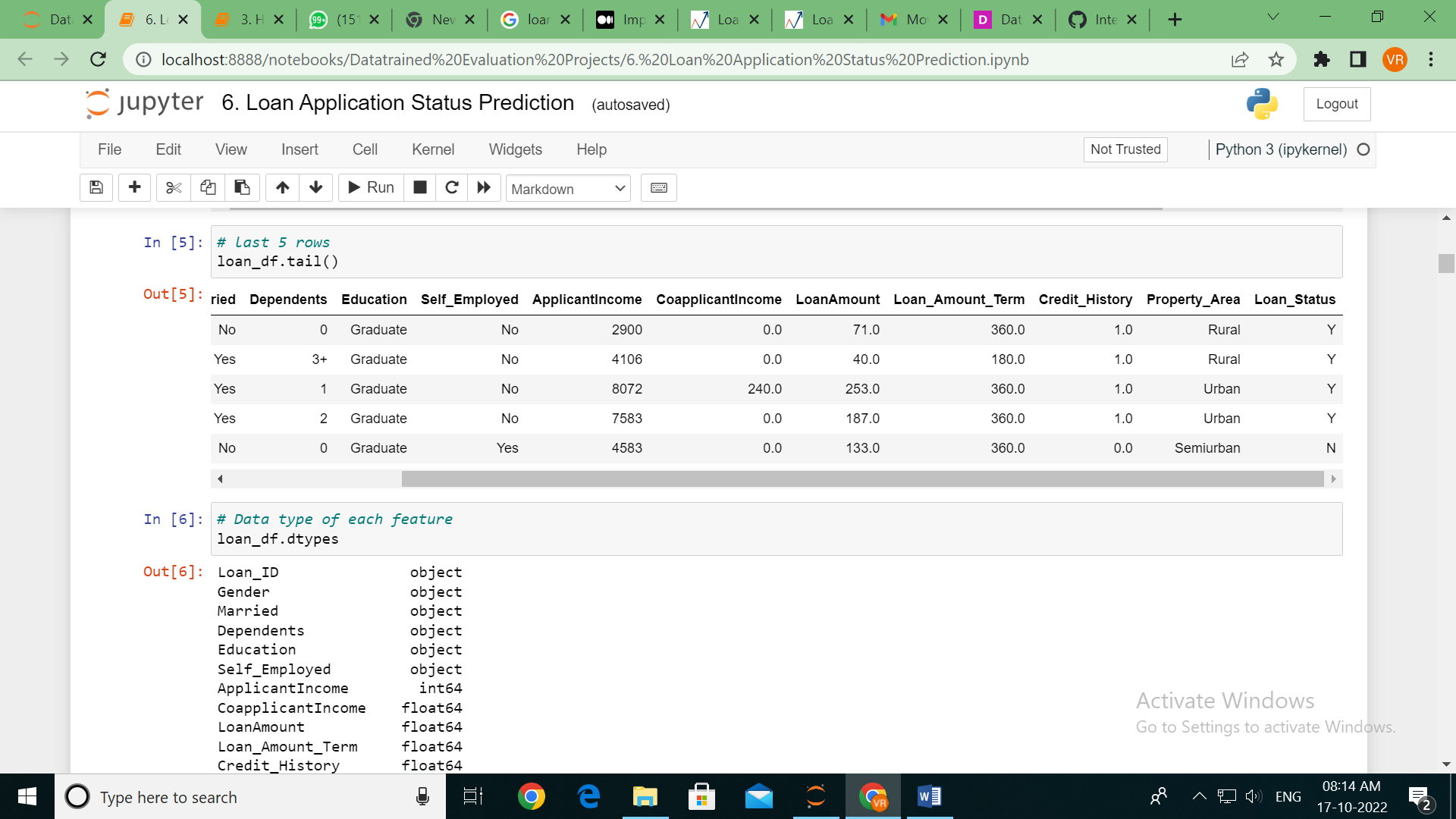
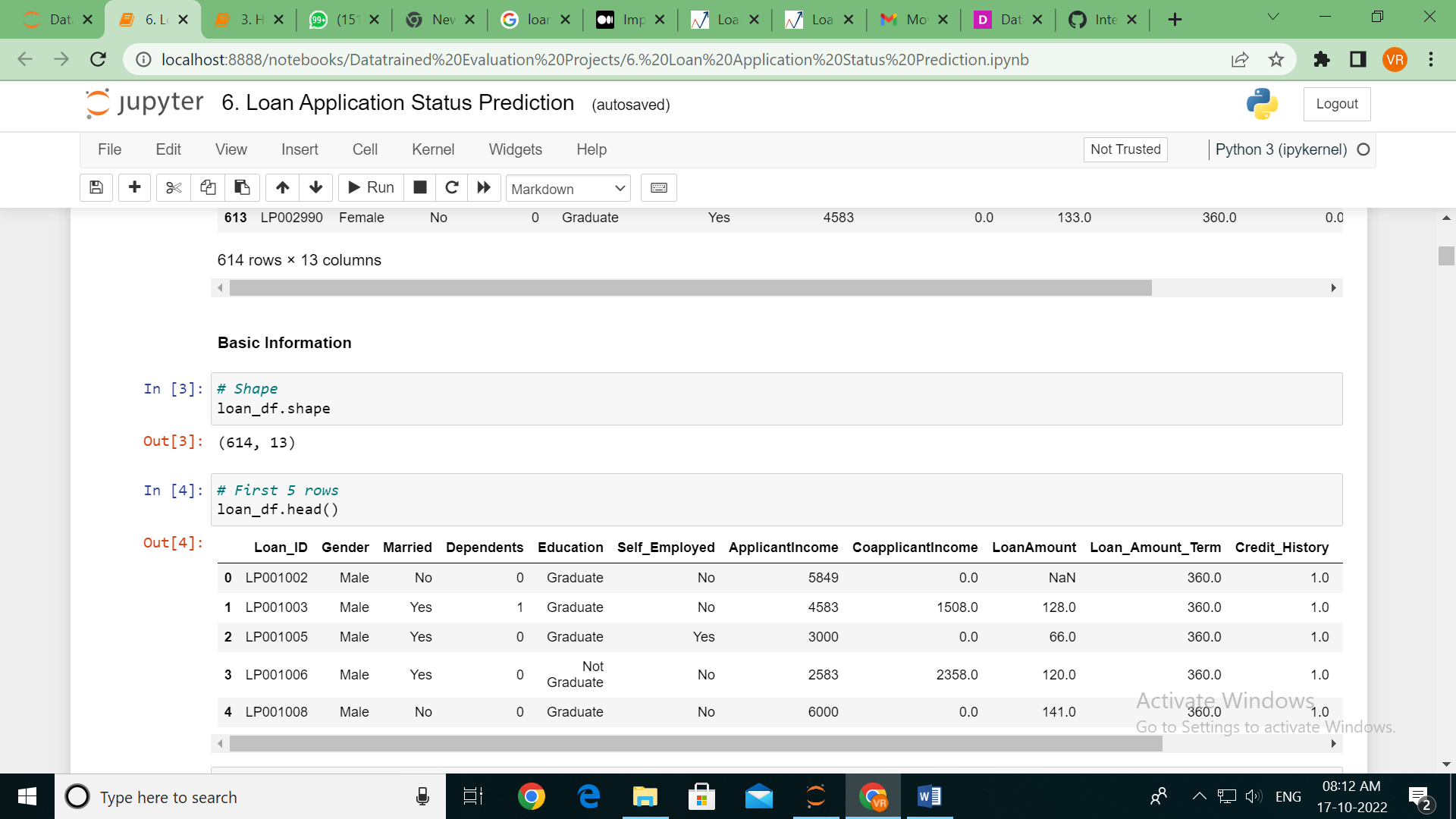
Code can be downloaded from - <https://github.com/vinayakr6/Internship/blob/main/Projects/6.%20Loan%20Application%20Status%20Prediction.ipynb>

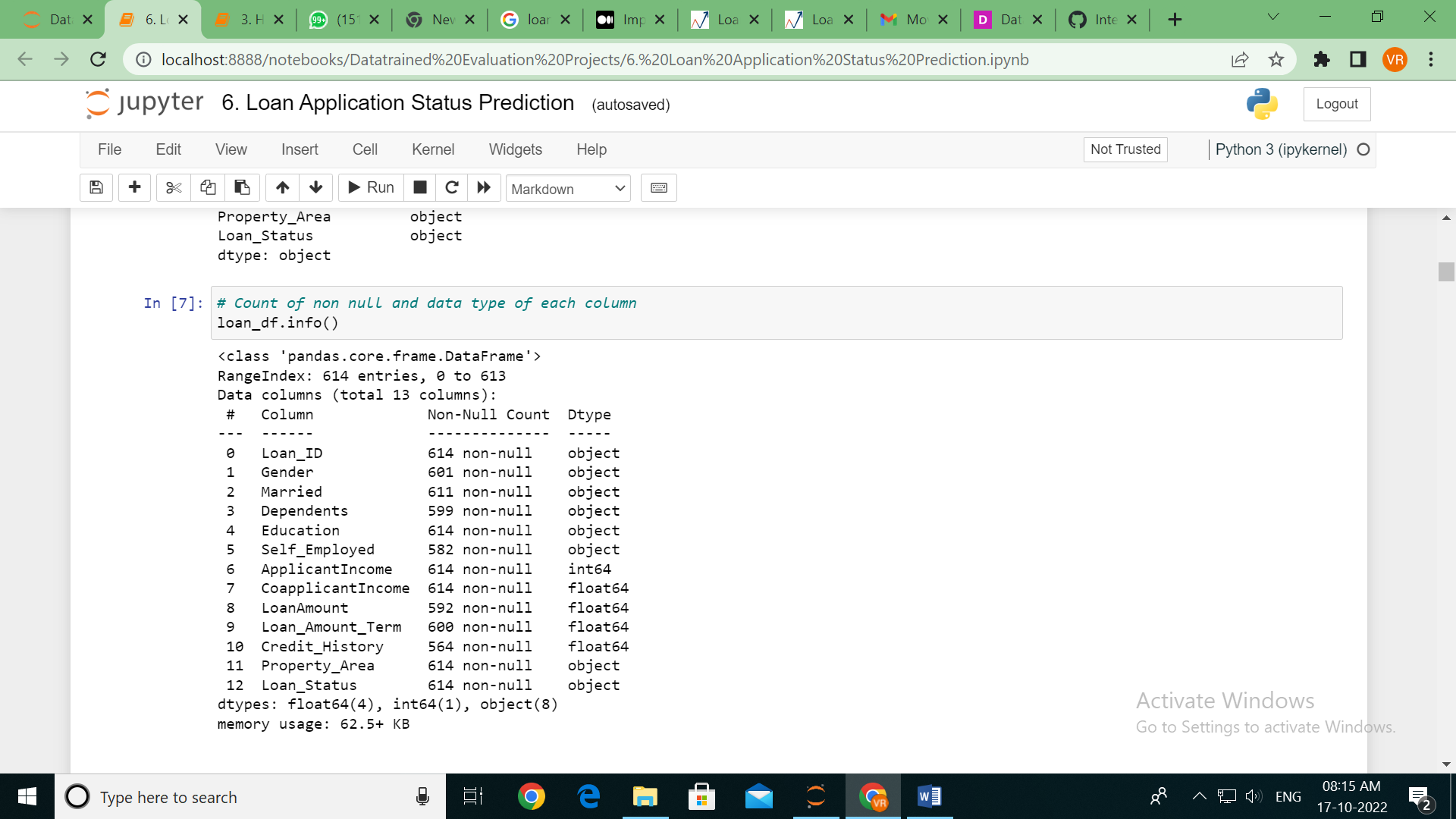
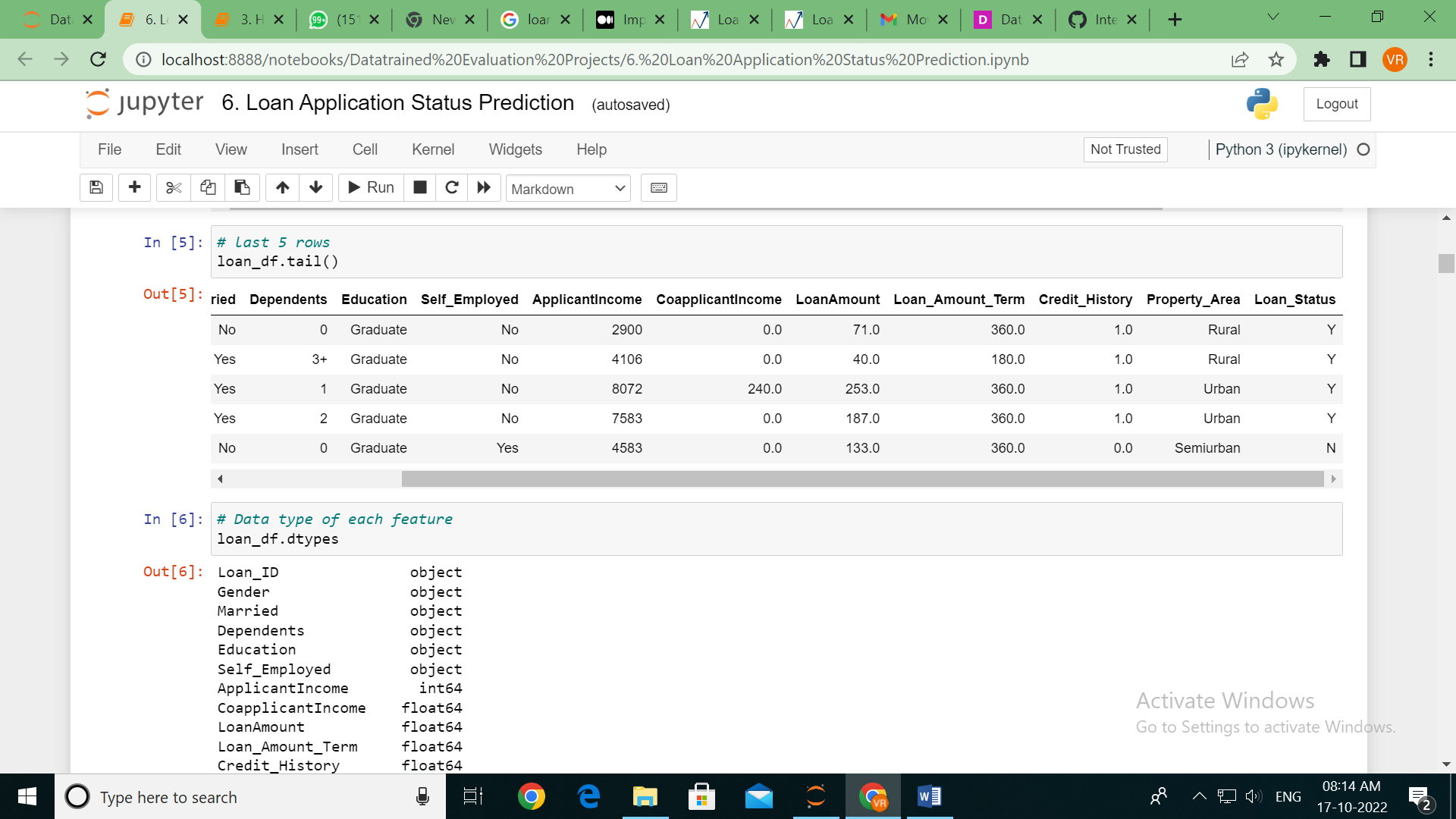
# **Importing Data**

Importing the libraries and the dataset



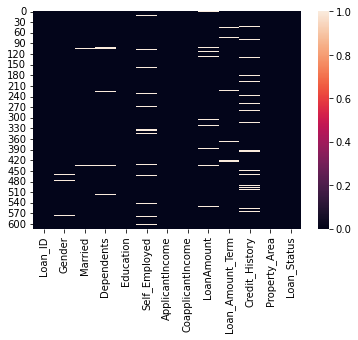
Basic information of the data like first five rows, last five rows, data types of all the features, shape of the dataset and info which provides us with non-null values and their datatypes present in the dataset.





# **Exploratory Data Analysis**

Checking for the null values – you can see that the null values are present in the data as given below along with the heat map of null values.



There are 'NaN' values present in the dataset. Before imputing the values with mean, median or mode or random value let's first check each features then decide on what to be imputed. For this a separate list of object datatype of numerical datatype is created.



Male 489

Female 112

Name: Gender, dtype: int64

Total values in Gender is 601 for total rows of 614

----------------------------

Yes 398

No 213

Name: Married, dtype: int64

Total values in Married is 611 for total rows of 614

----------------------------

0 345

1 102

2 101

3+ 51

Name: Dependents, dtype: int64

Total values in Dependents is 599 for total rows of 614

----------------------------

Graduate 480

Not Graduate 134

Name: Education, dtype: int64

Total values in Education is 614 for total rows of 614

----------------------------

No 500

Yes 82

Name: Self\_Employed, dtype: int64

Total values in Self\_Employed is 582 for total rows of 614

----------------------------

Semiurban 233

Urban 202

Rural 179

Name: Property\_Area, dtype: int64

Total values in Property\_Area is 614 for total rows of 614

----------------------------

Y 422

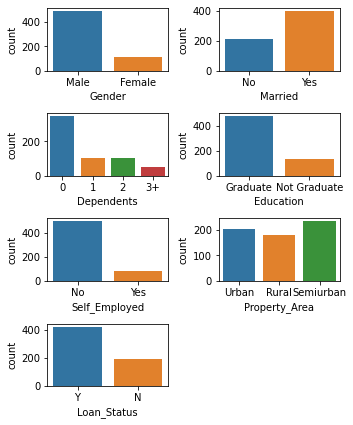
N 192

Name: Loan\_Status, dtype: int64

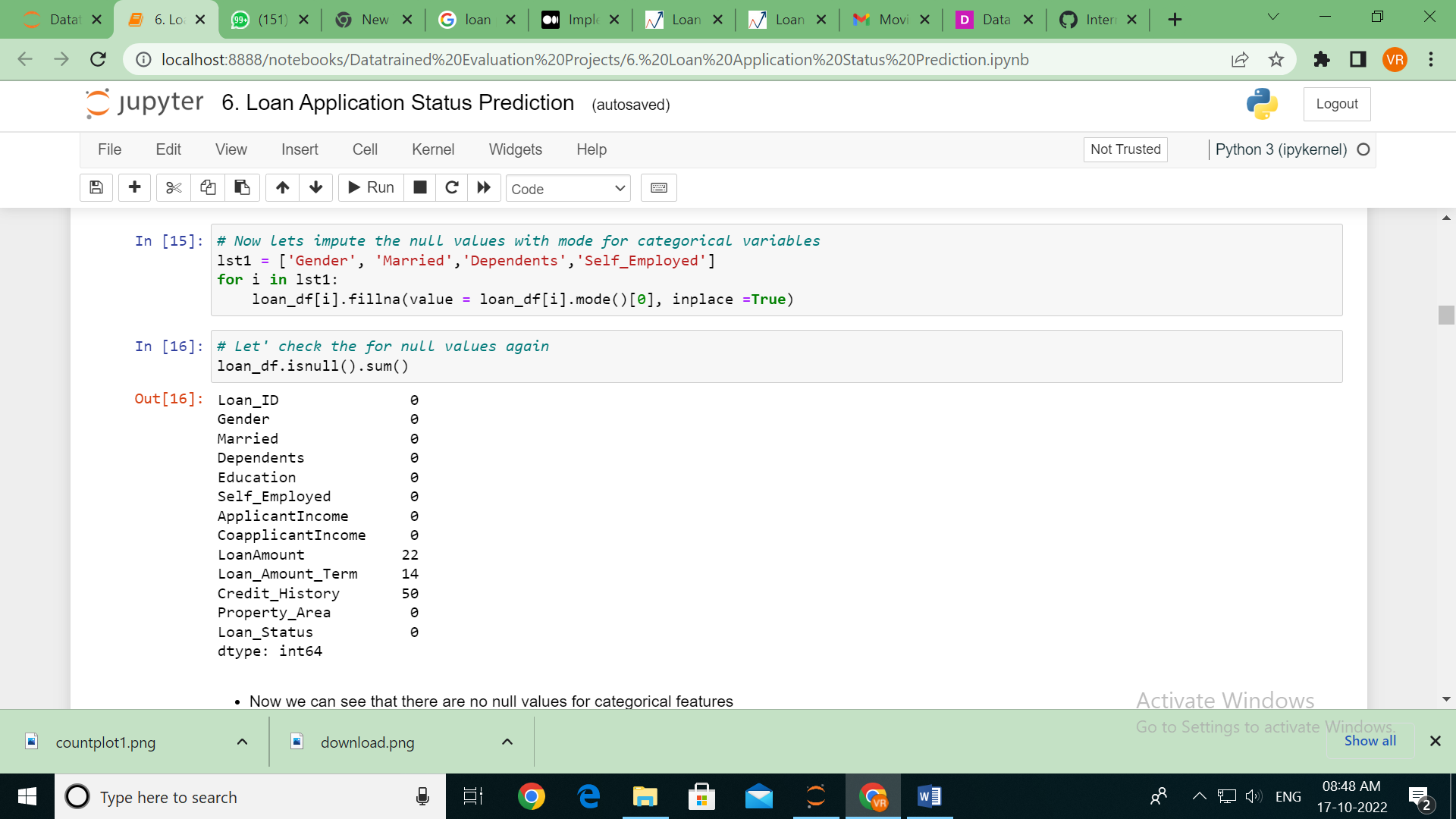
Total values in Loan\_Status is 614 for total rows of 614

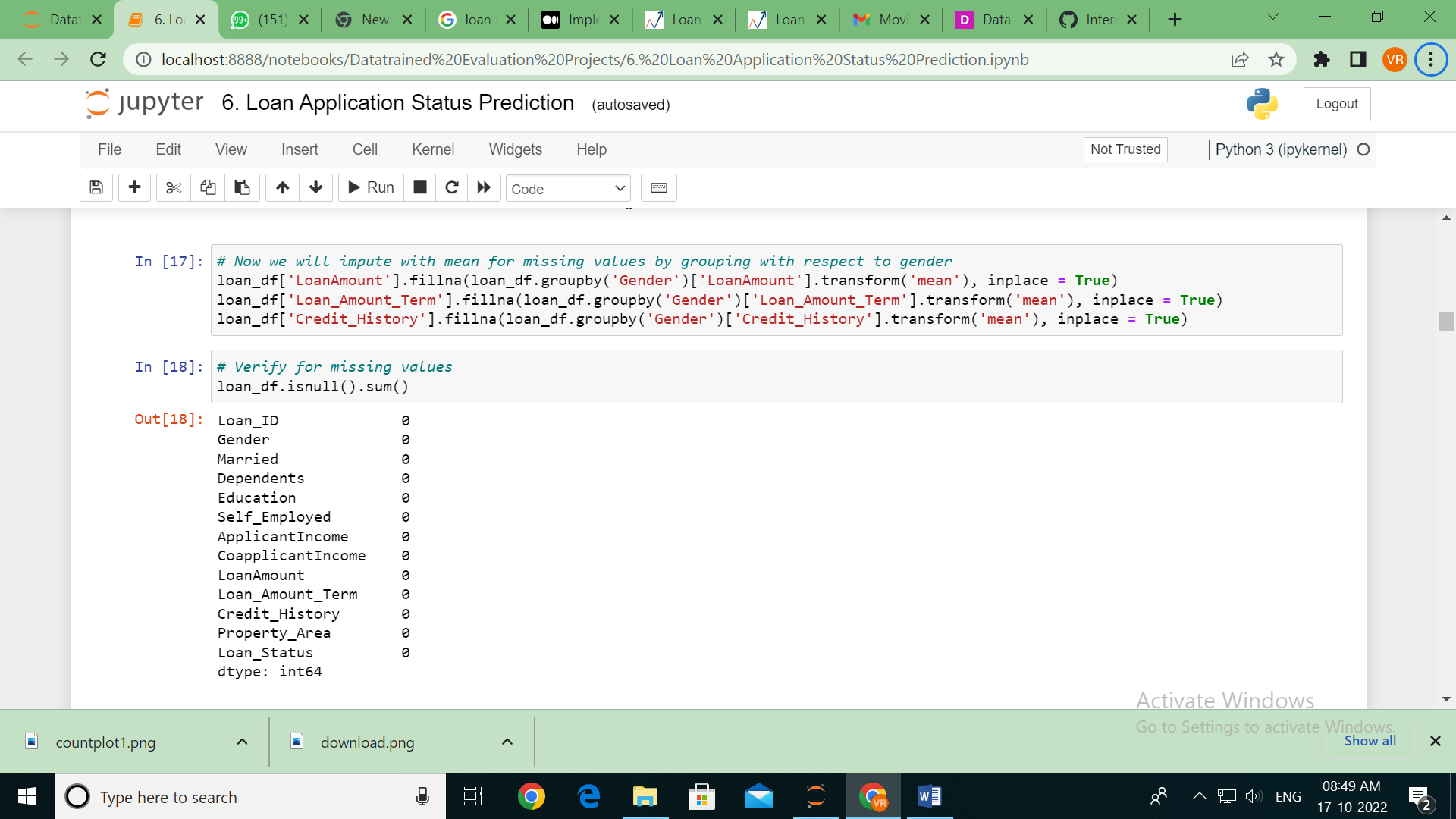
----------------------------

Count plot (Univariate plot) before imputing the missing values is shown below

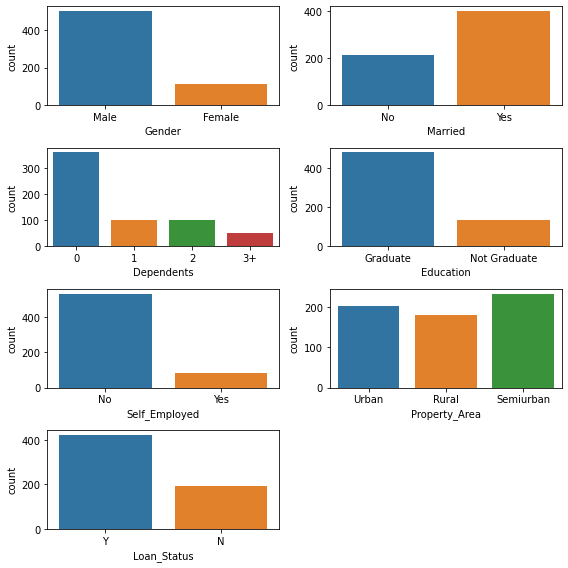


After plotting, the missing values in object datatype column is imputed with mode with most occurrence. Then, with isnull function verified the null values after imputation. Later, the numerical datatype columns were imputed with mean grouped with respect to gender and verified again to check whether any null values are present or not. The details are shown below





**Count plot (Univariate plot)** after imputing the missing values is shown below

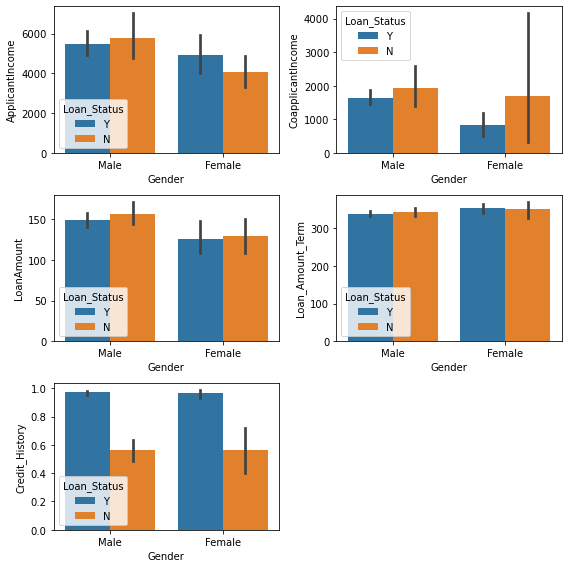


Observation for count plot

* More male than female
* More married people than unmarried
* many don't have dependents
* Most are graduate
* Few self employed
* Many have their loan approved than who don't.

**Bar plot (Bivariate analysis)**

1. Bar Plot for all numerical feature w.r.t 'Gender' and 'Loan\_Status’

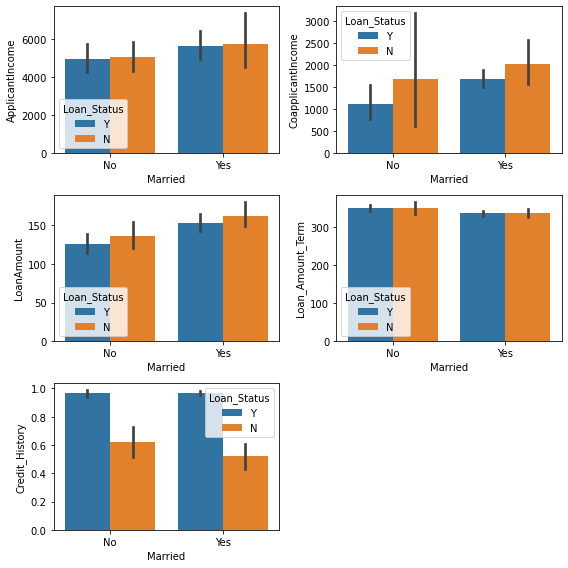


Observations

* For Male:
  + With Loan status yes income range 5000 to 6000, that with No have higher income from 5000 to 7000
* For Female:
  + With Loan status yes income range 4000 to 6000, that with No have less income from 3500 to 5000
* Coapplicants income without approval of loan have higher income range for both male and female
* Loan amount for male is higher than female
* Those with loan approved have higher credit history for both male and female

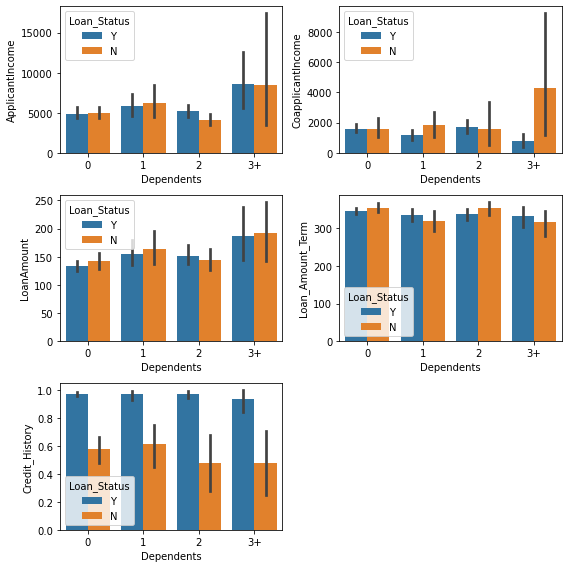
1. Bar Plot for all numerical feature w.r.t 'Married' and 'Loan\_Status'

* Applicants who are married have high income



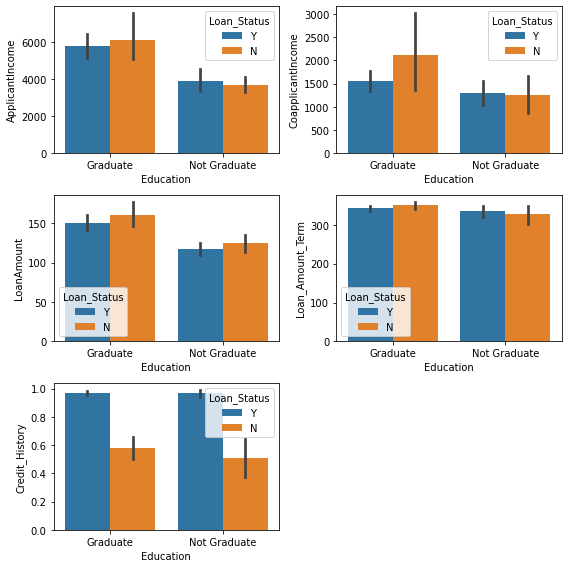
1. Bar Plot for all numerical feature w.r.t 'Dependents' and 'Loan\_Status'

* Applicants with more than 3 dependents have income both who's loan is approved or not and also same with co-applicant for non-approved loans



1. Bar Plot for all numerical feature w.r.t 'Education' and 'Loan\_Status'

* Graduated applicants have more income the not graduated and also the loan approval yes or no

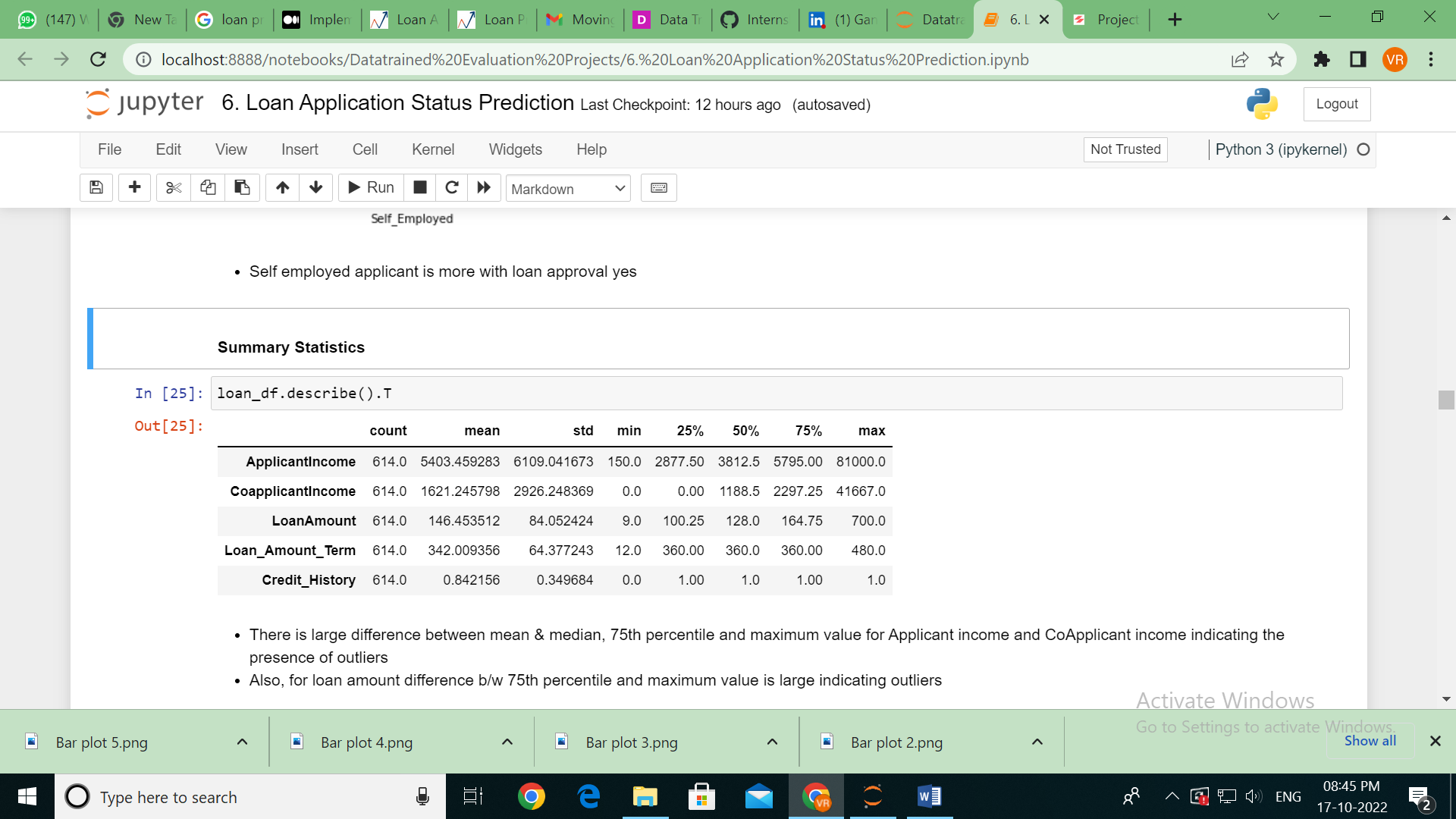


1. Bar Plot for all numerical feature w.r.t 'Self\_Employed' and 'Loan\_Status'

* Self-employed applicant is more with loan approval yes

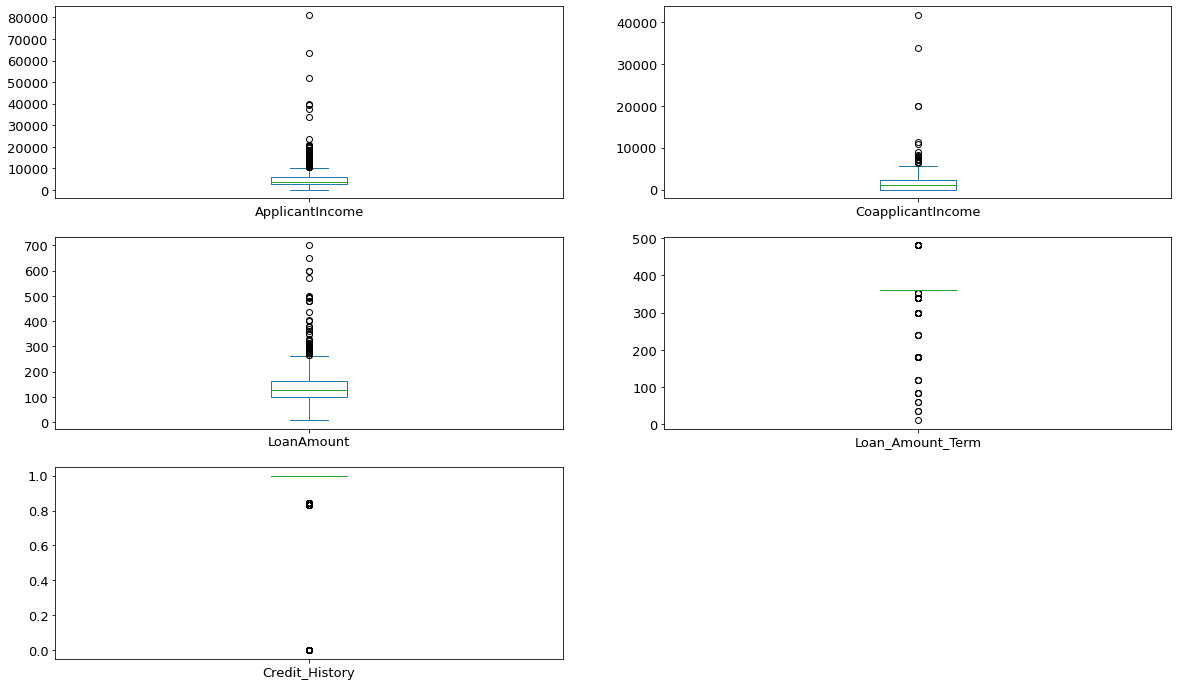


**Descriptive Statistics**



* Here is large difference between mean & median, 75th percentile and maximum value for Applicant income and Co-applicant income indicating the presence of outliers
* Also, for loan amount difference b/w 75th percentile and maximum value is large indicating outliers

**Outliers**



* Clearly see there are outliers present in all the numerical columns

Now, with the help of logical operators for different features with loan approval status yes is verified to know more about the data.

1. Let's see how many received loan approval with income greater than 5000

* loan\_df[(loan\_df["Loan\_Status"]=="Y") & (loan\_df['ApplicantIncome'] > 5000)]
* 132 people received approval

1. Let's see how many received loan approval with income greater than 5000 w.r.t. Gender

* loan\_df[(loan\_df["Loan\_Status"]=="Y") & (loan\_df['ApplicantIncome'] > 5000) & (loan\_df['Gender'] == 'Female')]
* Out of 132, 16 Female received loan approval and remaining Male

1. Let's see how many received loan approval with Graduation

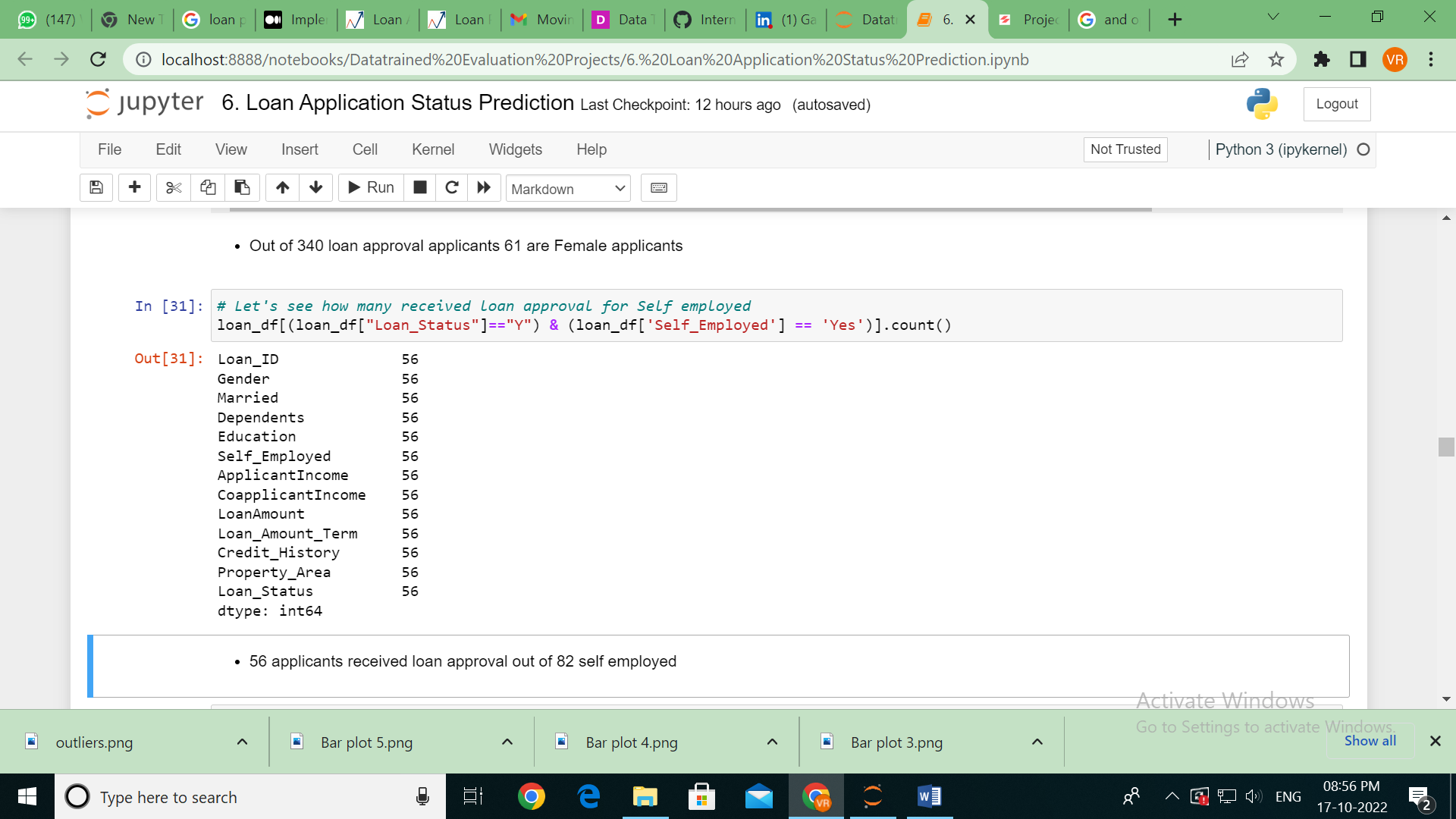
* loan\_df[(loan\_df["Loan\_Status"]=="Y") & (loan\_df['Education'] == 'Graduate')]
* Out of 480 graduates 340 got loan approval

1. Let's see how many received loan approval with Graduation w.r.t Gender

* loan\_df[(loan\_df["Loan\_Status"]=="Y") & (loan\_df['Education'] == 'Graduate') & (loan\_df['Gender'] == 'Female')]
* Out of 340 loan approval applicants 61 are Female applicants

1. Let's see how many received loan approval for Self employed

* loan\_df[(loan\_df["Loan\_Status"] == "Y") & (loan\_df['Self\_Employed'] == 'Yes')].count()
* 56 applicants received loan approval out of 82 self-employed as shown below



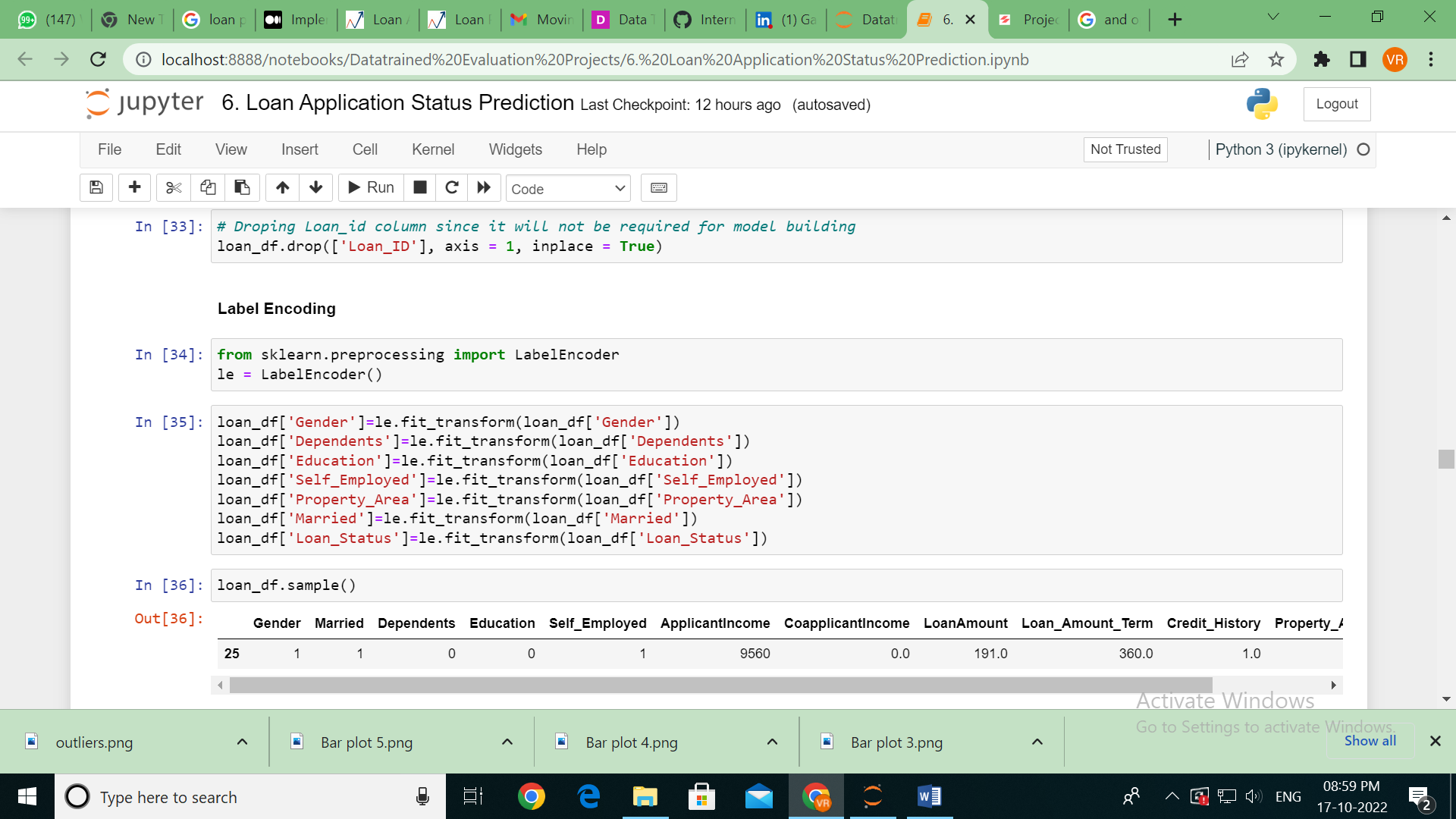
1. Let's see how many received loan approval who are not Self employed

* loan\_df[(loan\_df["Loan\_Status"]=="Y") & (loan\_df['Self\_Employed'] == 'No')].count()
* Out of 500 not self-employed 366 applicants received loan approval.

Dropping Loan\_id column since it will not be required for model building

* loan\_df.drop(['Loan\_ID'], axis = 1, inplace = True)

**Label Encoding**



All the object datatype columns are converted to numerical using label encoding as shown above.

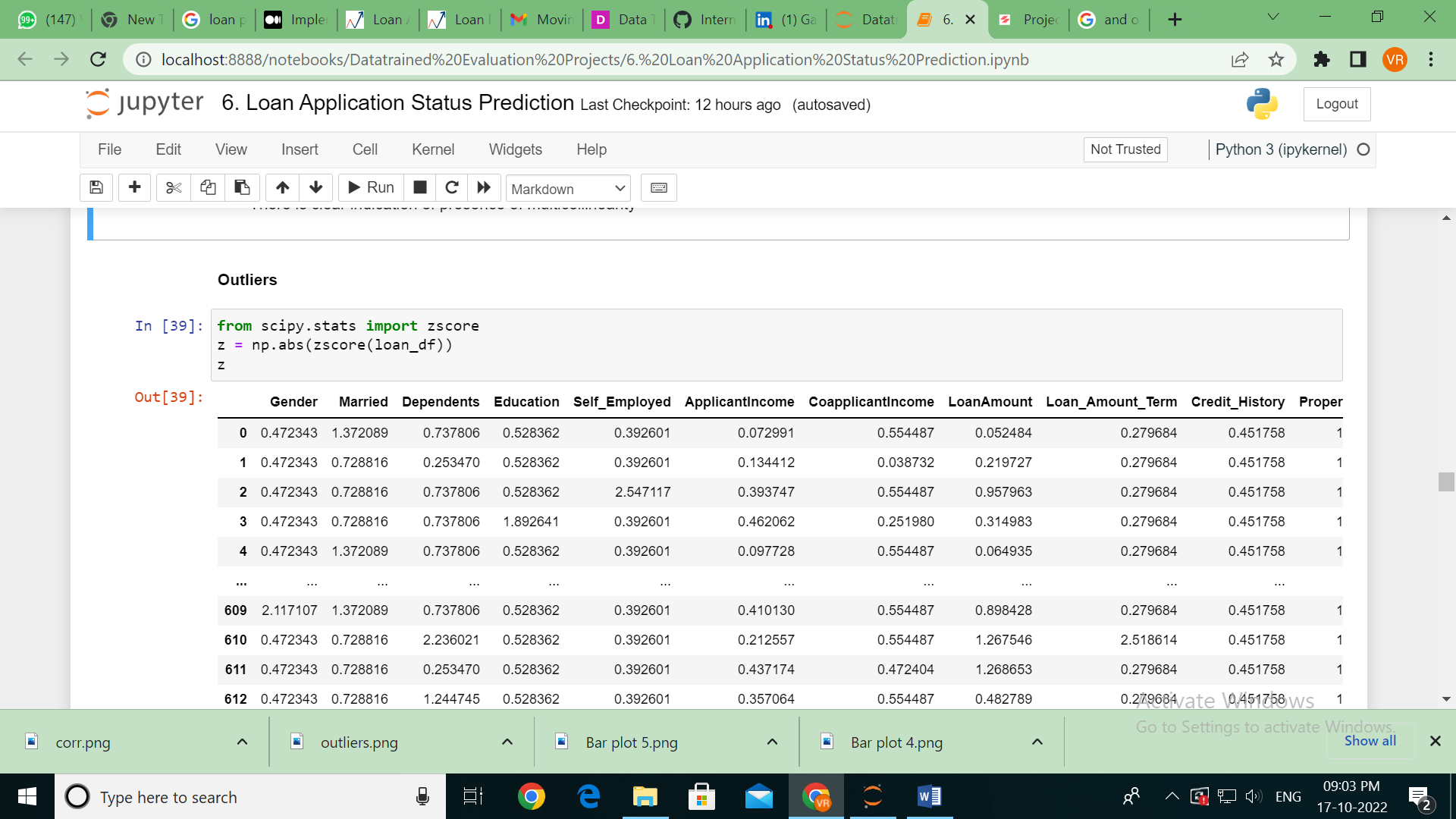
**Correlation**

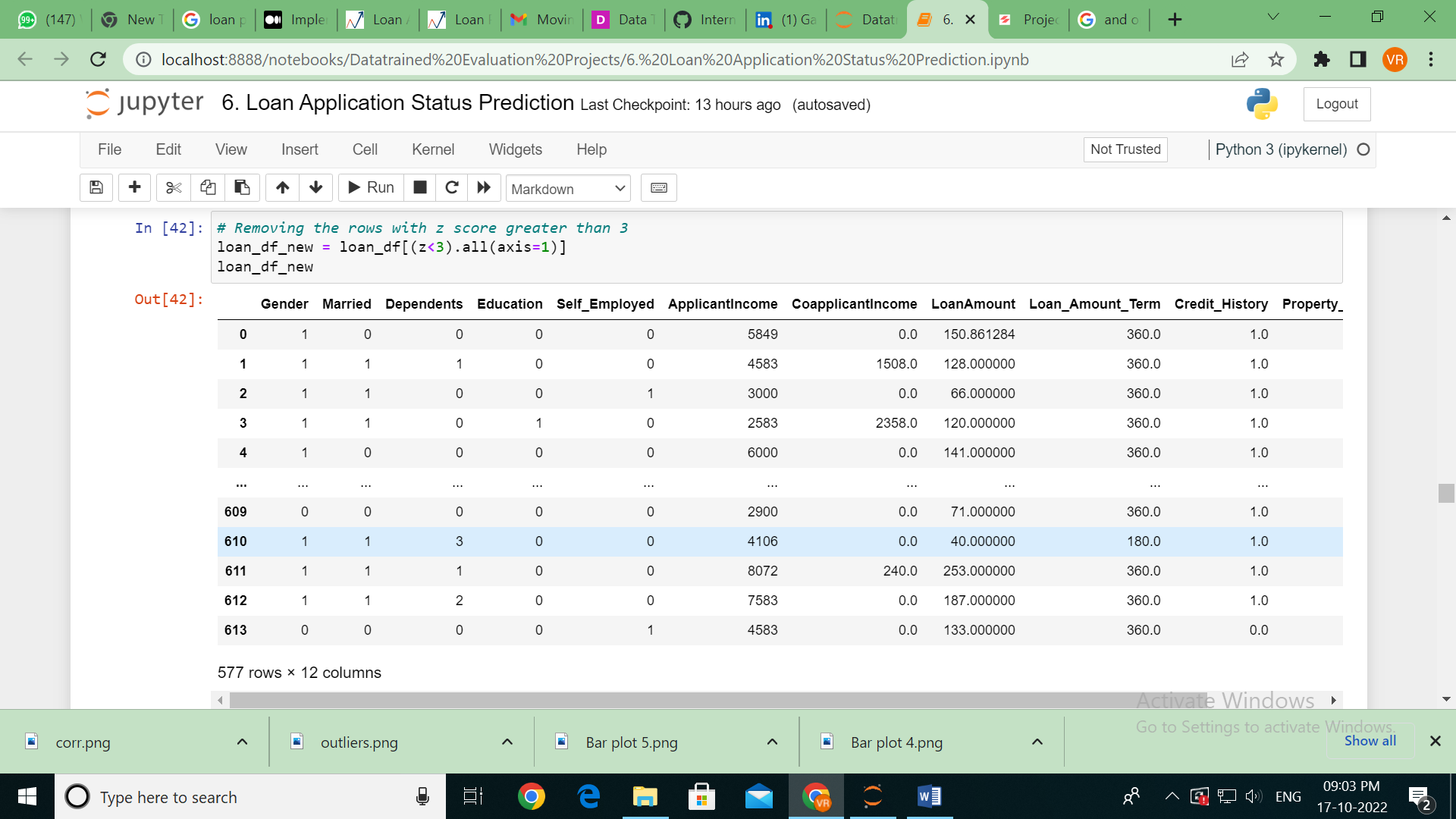
Credit history is highly correlated with target variable

There is clear indication of presence of multicollinearity

****

**Removing outliers using Z-score**





Checking the shape of the dataset before and after removing outliers

print("Old DataFrame:-", loan\_df.shape)

print("New DataFrame:-", loan\_df\_new.shape)

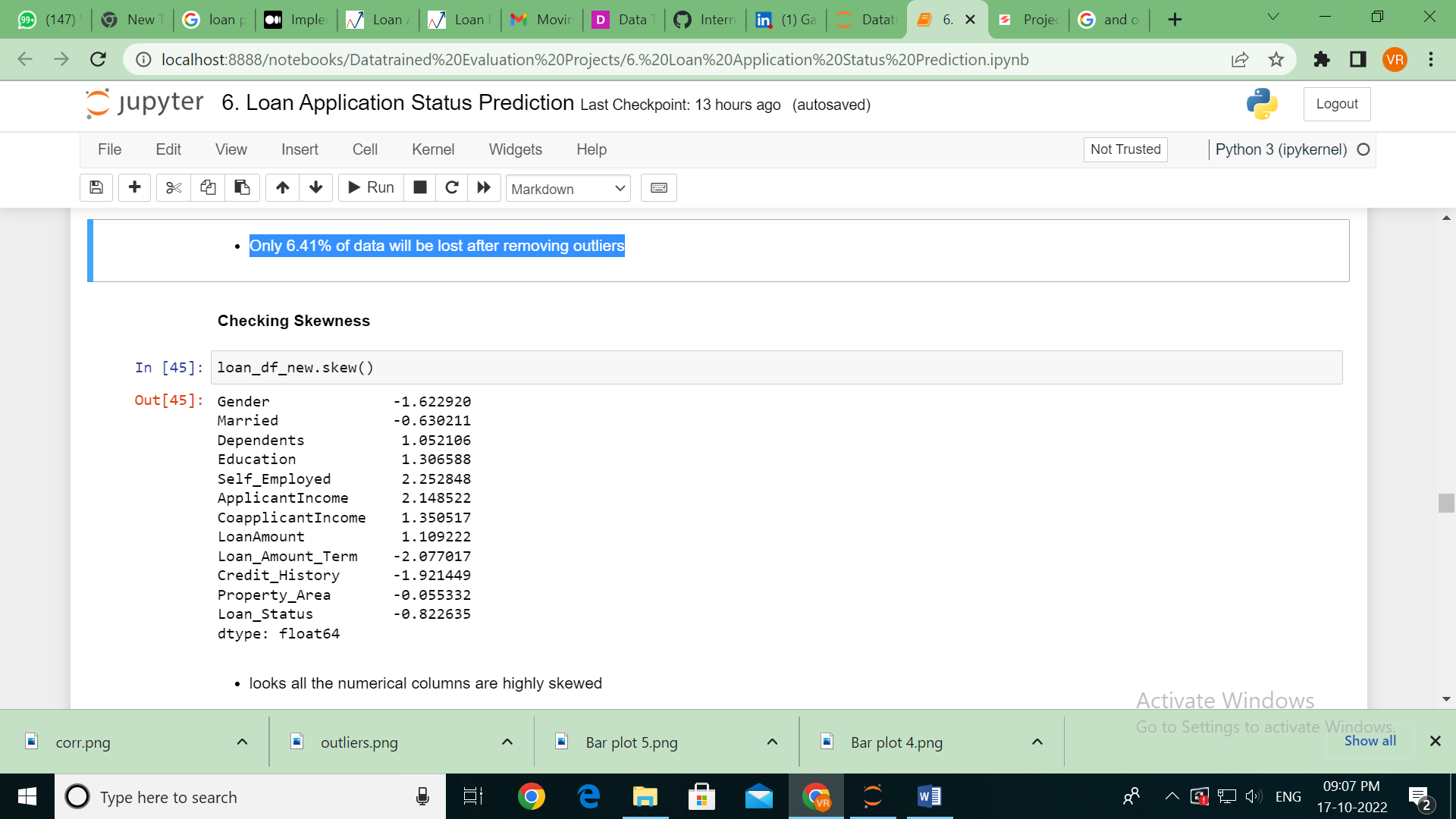
print('Total dropped rows:-', loan\_df.shape[0]-loan\_df\_new.shape[0])

Old DataFrame:- (614, 12)

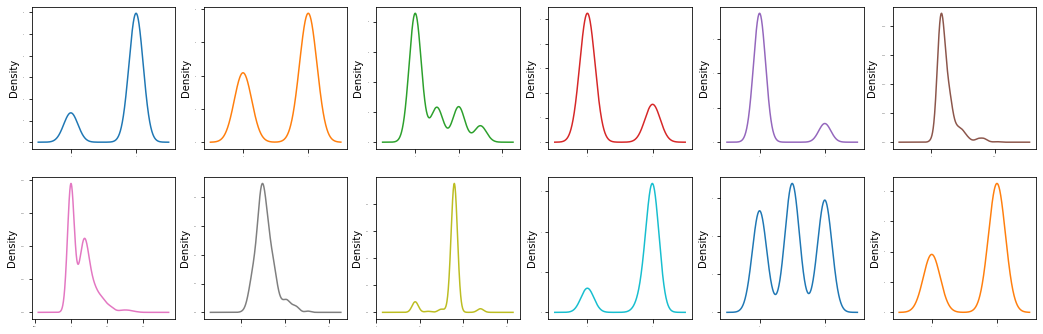
New DataFrame:- (577, 12)

Total dropped rows:- 37 in the dataset after removing the outliers which amounts to 6.41% of data which is less than 10%. Hence, we can proceed further with the new dataset considered after removing the outliers.

**Skewness**



Looks all the numerical columns are highly skewed. Below distribution plot shows the skewness whether it is right skewed/left skewed/ normally distributed.



Skewness is removed using transformation, values after removing skewness are given below.

for index in X.skew().index:

if X.skew().loc[index]>0:

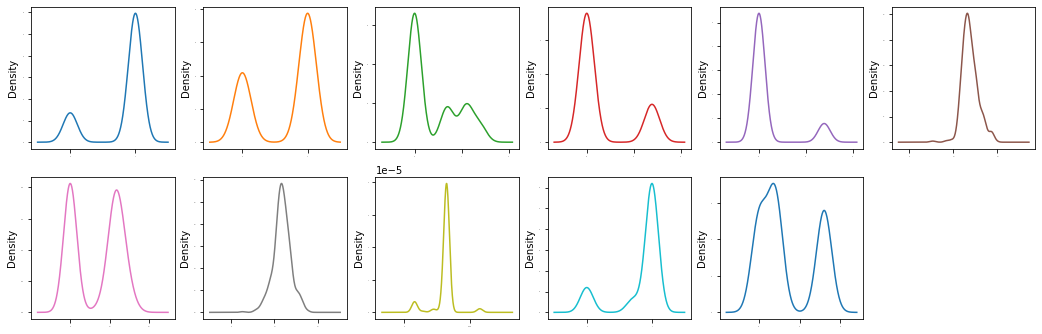
X[index]=np.log1p(X[index])

if X.skew().loc[index]<0:

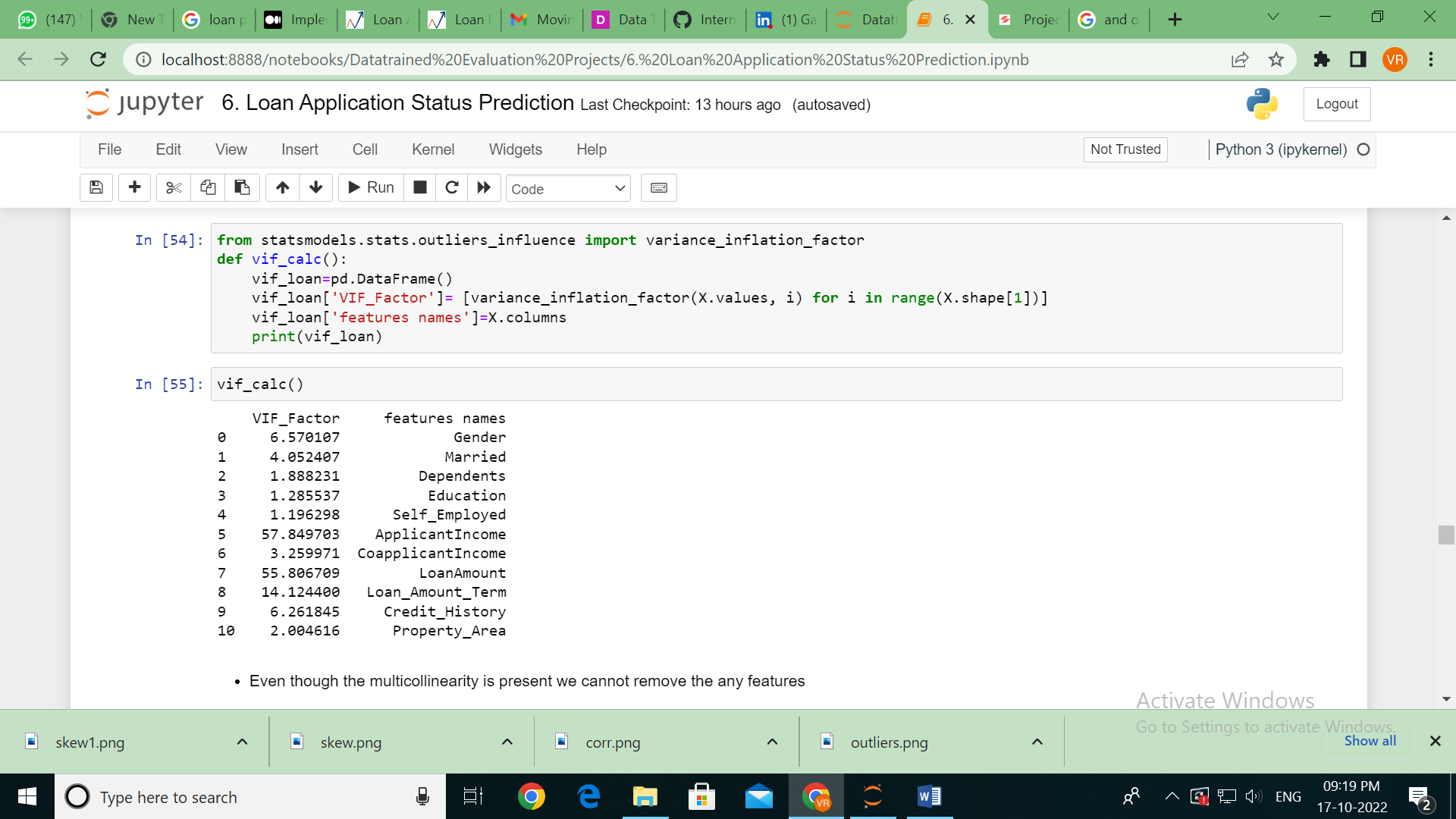
X[index]=np.square(X[index])



Distribution plot after skewness is removed



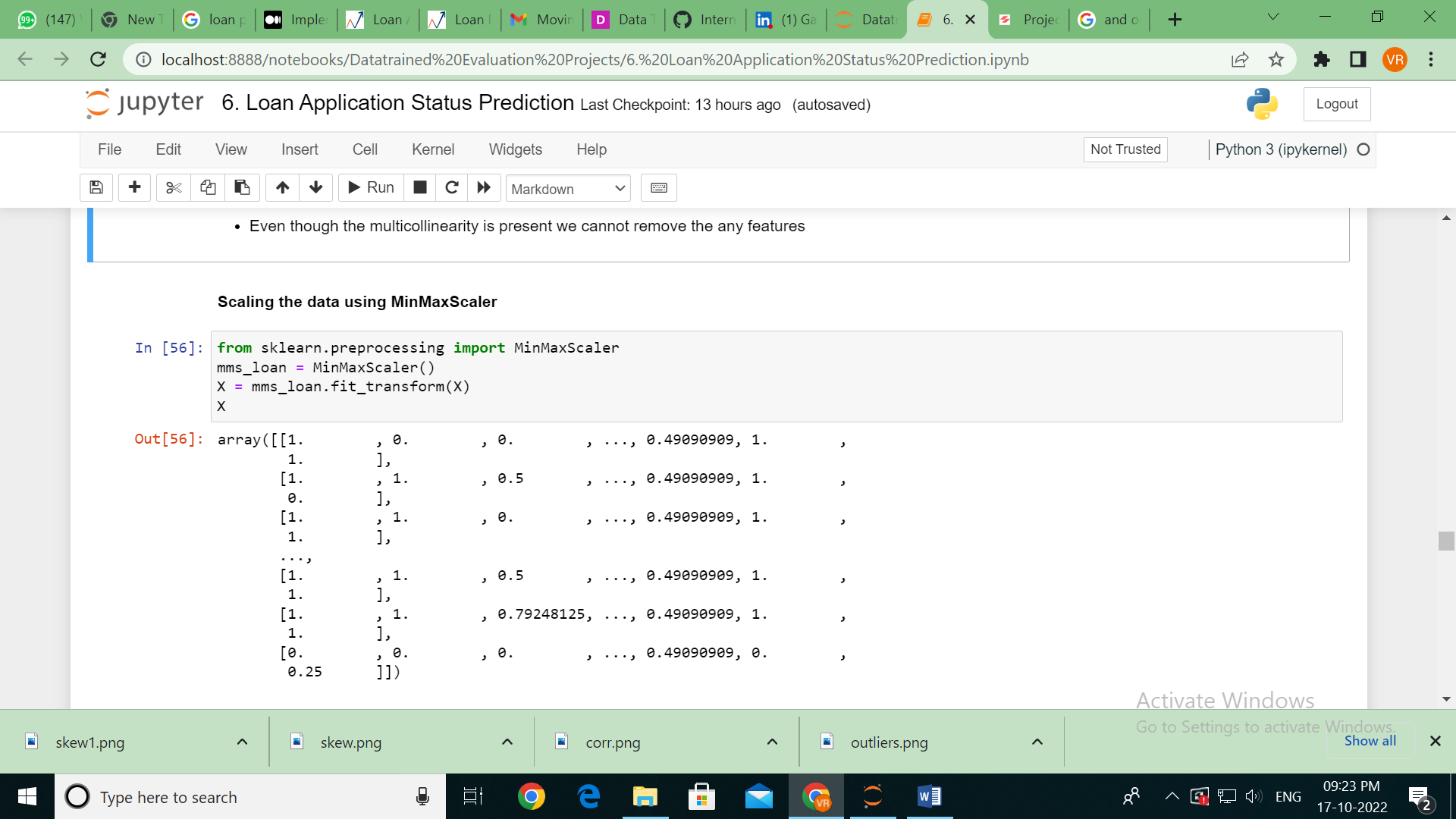
Variance Inflation factor



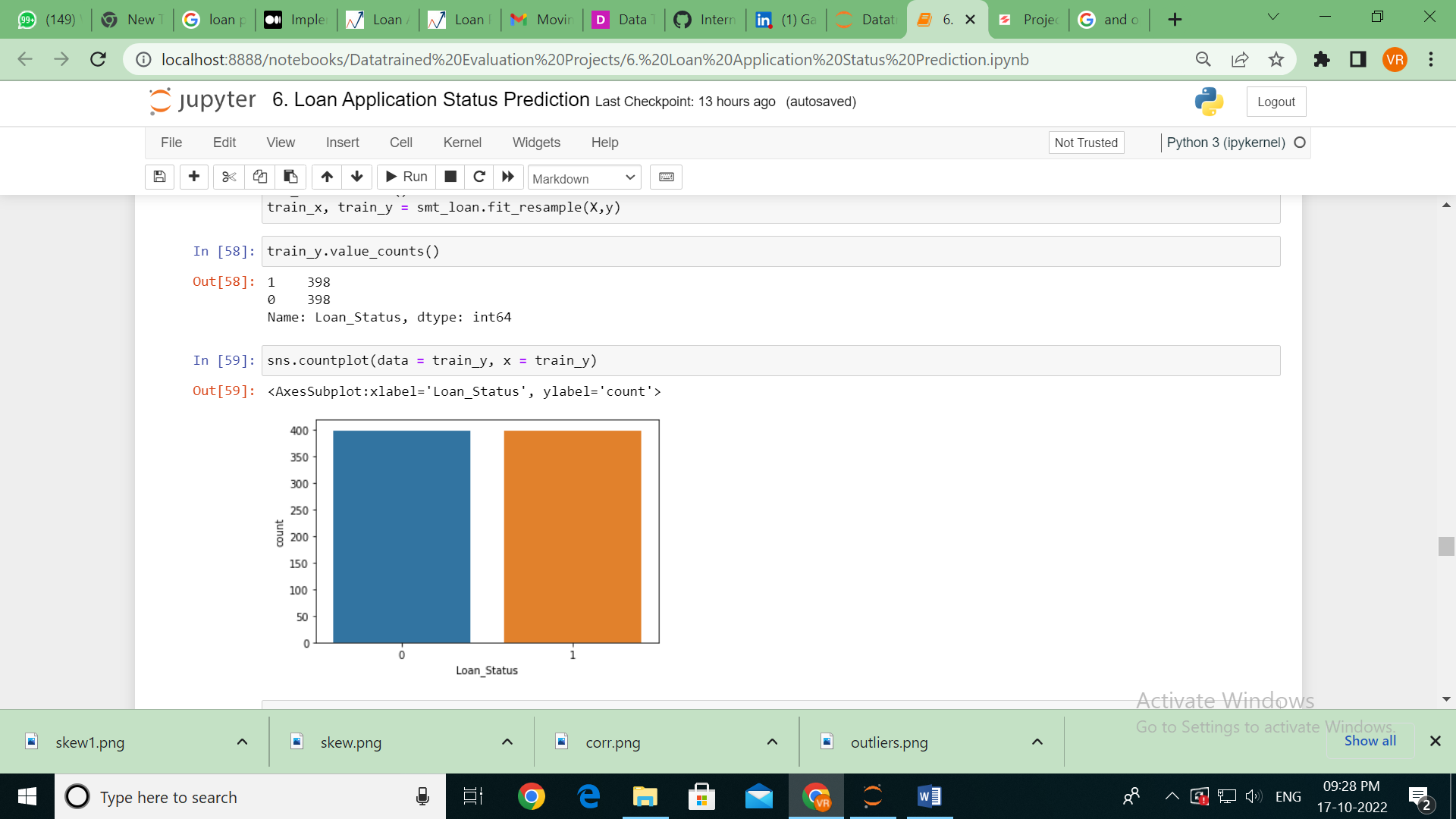
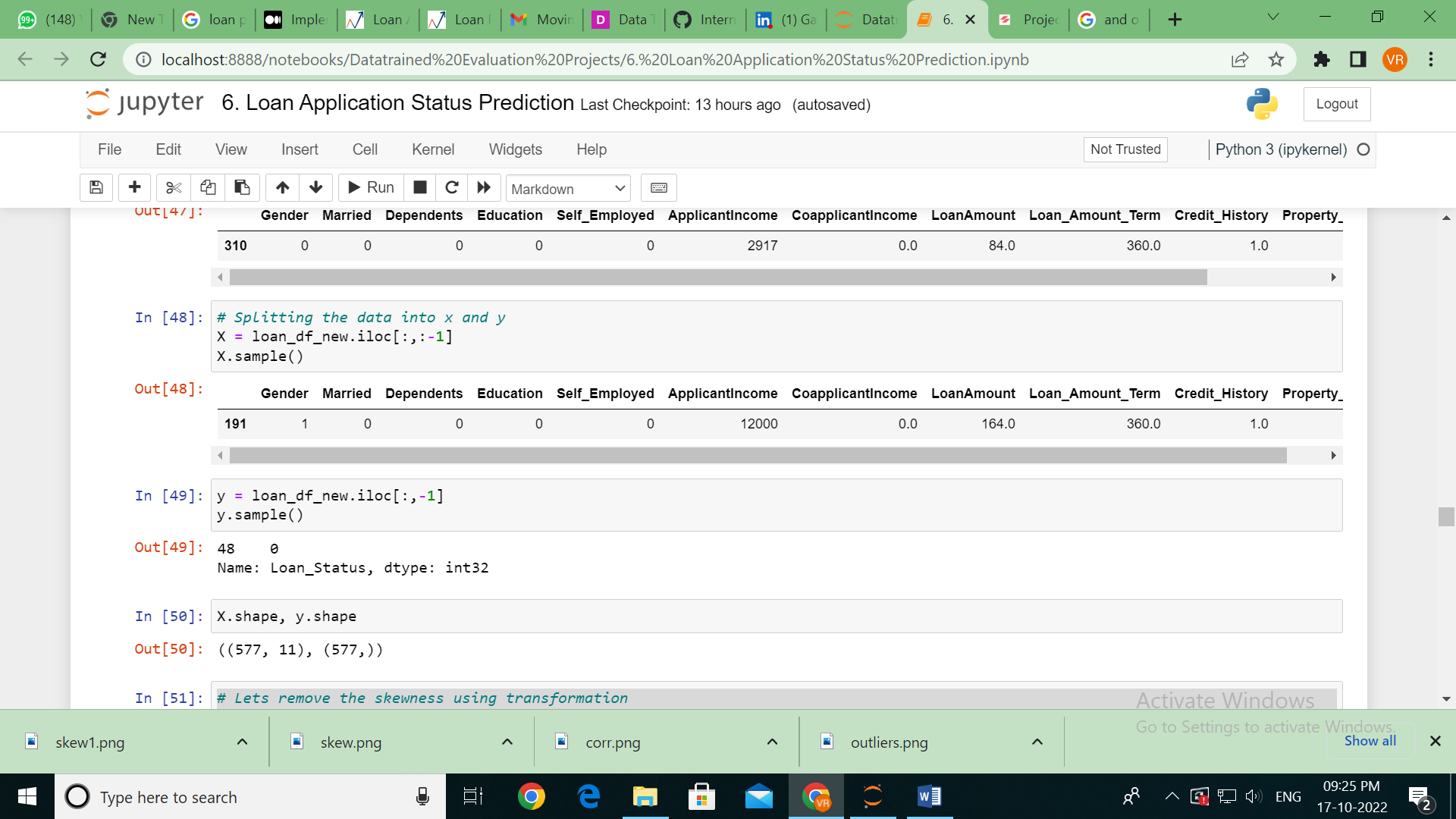
Even though the multi-collinearity is present we cannot remove the any features further because we have already lost 6.41% data after removing the outliers.

**Model Building**

Before applying various algorithms for predicted the loan status the data is scaled using MinMaxScaler.

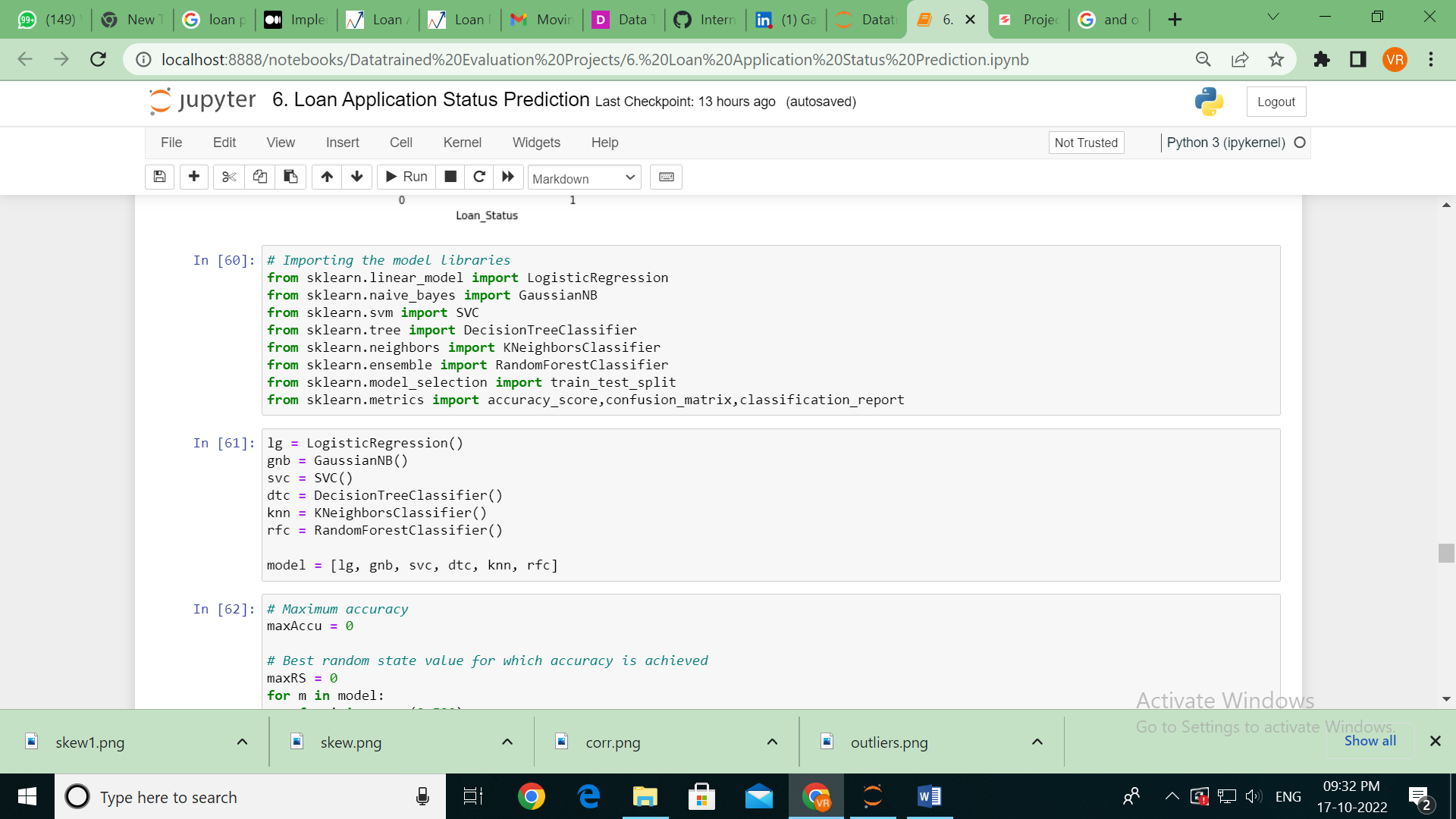


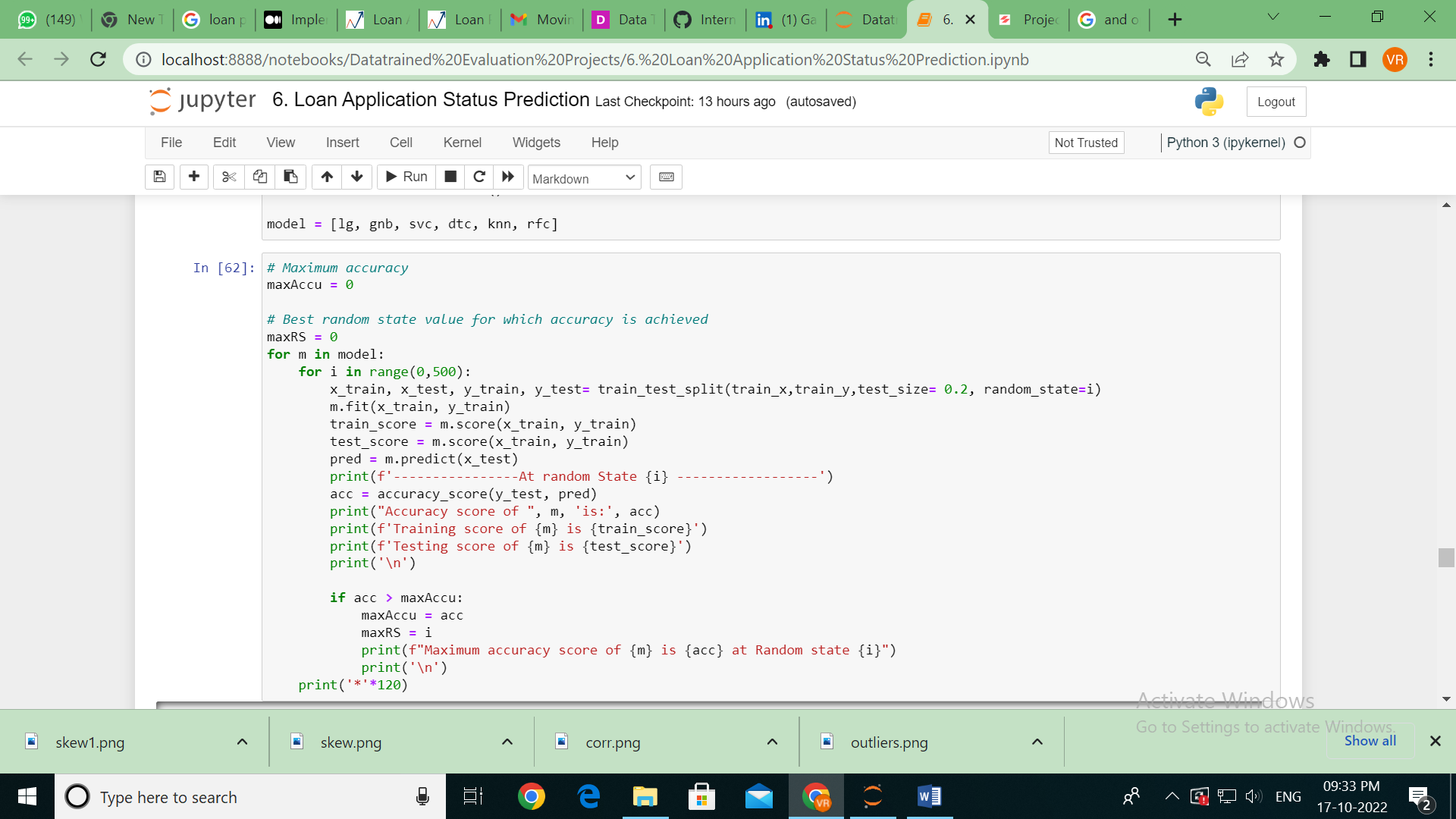
Since the loan status (target variable) feature is imbalanced, SMOTE is applied before training the data with various models. Before this the data had to be split into X (independent variables) and y (dependent variable) which has been done before checking for skewness as shown below.



Now, we can see that after SMOTE the data is balanced.

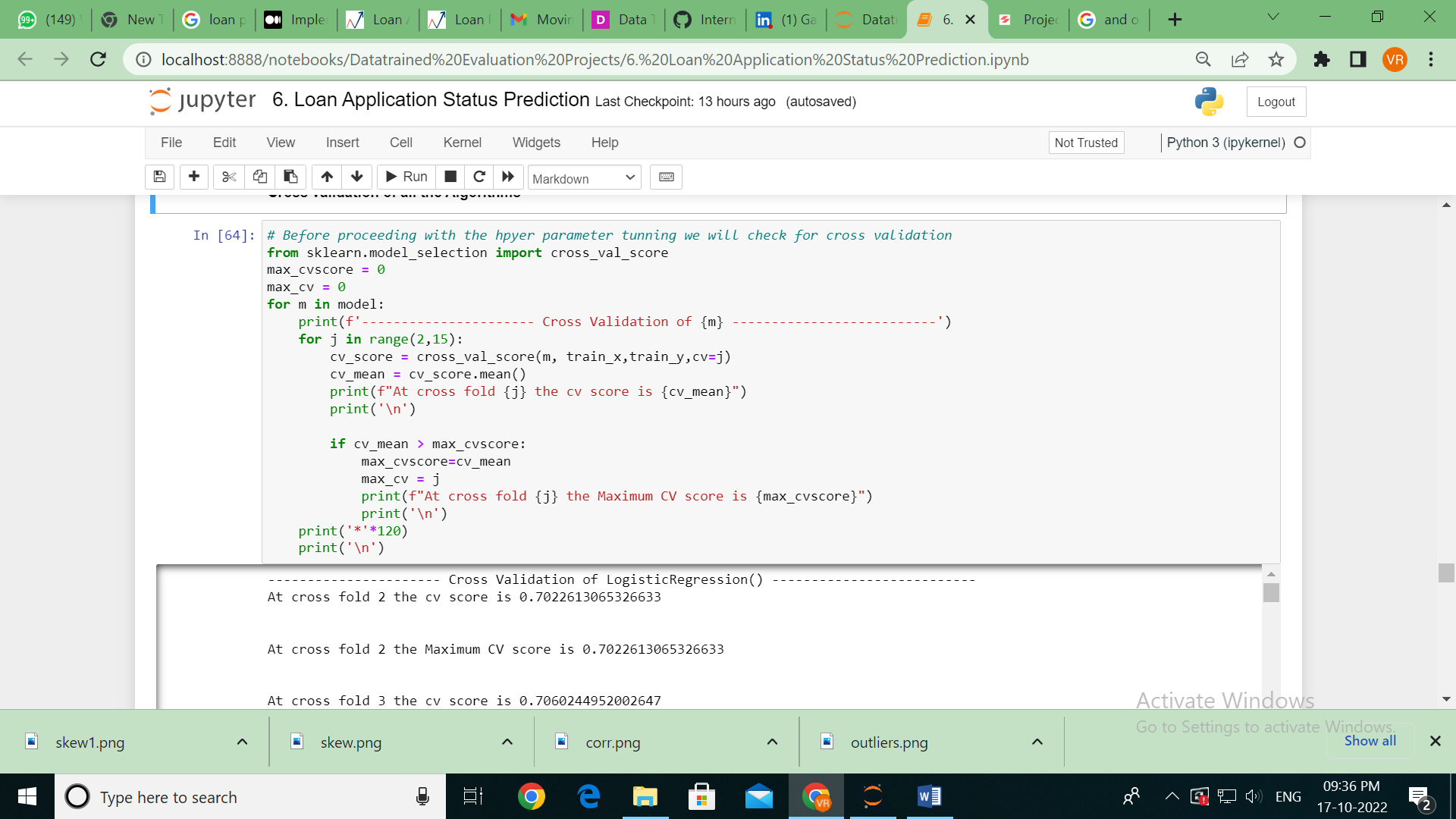
Importing libraries for model building as shown below. The is modelled using 6 models from training to get best random state in the random state range of 0 to 500. Next, cross validation is done for all the models to get the best cv.





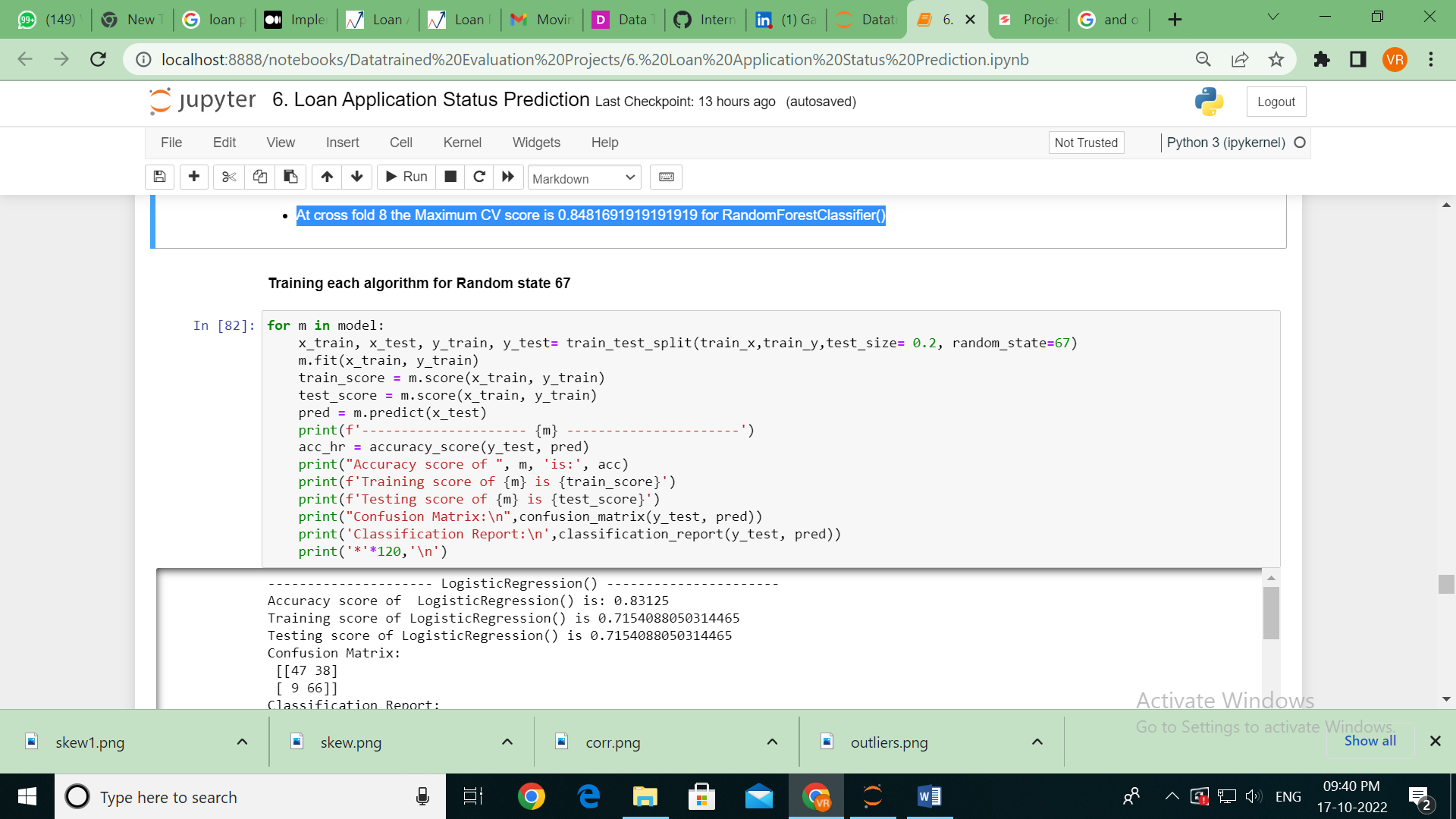
* Getting the Maximum accuracy score of Random Forest Classifier is 0.9 at Random state 67

Cross validation of all the Algorithms



* At cross fold 8 the Maximum CV score is 0.8481 for Random Forest Classifier

Now, for a random state of 67 all the models are trained and following results were obtained accuracy score, training and testing score, confusion matrix and classification report and the results for each model is shown below.



1. **Logistic Regression**

Accuracy score of LogisticRegression() is: 0.83125

Training score of LogisticRegression() is 0.7154088050314465

Testing score of LogisticRegression() is 0.7154088050314465

Confusion Matrix:

[[47 38]

[ 9 66]]

Classification Report:

precision recall f1-score support

0 0.84 0.55 0.67 85

1 0.63 0.88 0.74 75

accuracy 0.71 160

macro avg 0.74 0.72 0.70 160

weighted avg 0.74 0.71 0.70 160

1. **Gaussian Naïve Bayes classifier**

Accuracy score of GaussianNB() is: 0.83125

Training score of GaussianNB() is 0.7279874213836478

Testing score of GaussianNB() is 0.7279874213836478

Confusion Matrix:

[[44 41]

[ 9 66]]

Classification Report:

precision recall f1-score support

0 0.83 0.52 0.64 85

1 0.62 0.88 0.73 75

accuracy 0.69 160

macro avg 0.72 0.70 0.68 160

weighted avg 0.73 0.69 0.68 160

1. **Support Vector Classifier**

Accuracy score of SVC() is: 0.83125

Training score of SVC() is 0.7547169811320755

Testing score of SVC() is 0.7547169811320755

Confusion Matrix:

[[48 37]

[ 9 66]]

Classification Report:

precision recall f1-score support

0 0.84 0.56 0.68 85

1 0.64 0.88 0.74 75

accuracy 0.71 160

macro avg 0.74 0.72 0.71 160

weighted avg 0.75 0.71 0.71 160

1. **Decision Tree Classifier**

Accuracy score of DecisionTreeClassifier() is: 0.83125

Training score of DecisionTreeClassifier() is 1.0

Testing score of DecisionTreeClassifier() is 1.0

Confusion Matrix:

[[68 17]

[20 55]]

Classification Report:

precision recall f1-score support

0 0.77 0.80 0.79 85

1 0.76 0.73 0.75 75

accuracy 0.77 160

macro avg 0.77 0.77 0.77 160

weighted avg 0.77 0.77 0.77 160

1. **K Neighbors Classifier**

Accuracy score of KNeighborsClassifier() is: 0.83125

Training score of KNeighborsClassifier() is 0.8176100628930818

Testing score of KNeighborsClassifier() is 0.8176100628930818

Confusion Matrix:

[[59 26]

[25 50]]

Classification Report:

precision recall f1-score support

0 0.70 0.69 0.70 85

1 0.66 0.67 0.66 75

accuracy 0.68 160

macro avg 0.68 0.68 0.68 160

weighted avg 0.68 0.68 0.68 160

1. **Random Forest Classifier**

Accuracy score of RandomForestClassifier() is: 0.83125

Training score of RandomForestClassifier() is 1.0

Testing score of RandomForestClassifier() is 1.0

Confusion Matrix:

[[72 13]

[ 9 66]]

Classification Report:

precision recall f1-score support

0 0.89 0.85 0.87 85

1 0.84 0.88 0.86 75

accuracy 0.86 160

macro avg 0.86 0.86 0.86 160

weighted avg 0.86 0.86 0.86 160

One can clearly see that Random Forest is giving better results when we see at accuracy, Confusion matrix and Classification report.

**Hyper parameter tuning**

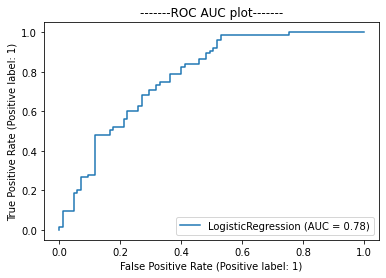
Hyper parameter tuning is applied to all the models using Grid Search CV and later for each model ROC AUC plot is plotted. Below are best parameters and estimators along with the final accuracy for each model along with roc auc plot.

1. **Logistic Regression**

Best parameters: {'C': 0.01, 'multi\_class': 'multinomial', 'penalty': 'l2', 'solver': 'newton-cg'}

Best Estimator: LogisticRegression(C=0.01, multi\_class='multinomial', solver='newton-cg')

Final Accuracy with Logistic Regression: 0.69375

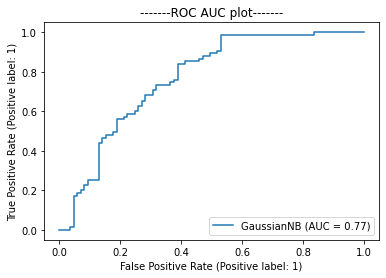


1. **Gaussian Naïve Bayes classifier**

Best parameters: {'var\_smoothing': 0.0008111308307896872}

Best Estimator: GaussianNB(var\_smoothing=0.0008111308307896872)

Final Accuracy with Gaussian Naive Bayes: 0.6875

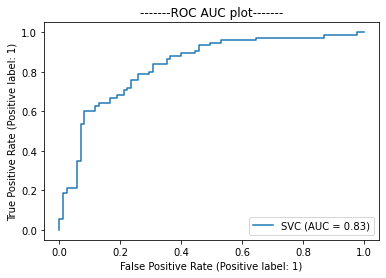


1. **Support Vector Classifier**

Best parameters: {'C': 10, 'kernel': 'poly'}

Best Estimator: SVC(C=10, kernel='poly')

Final Accuracy with Support Vector Classifier: 0.7625

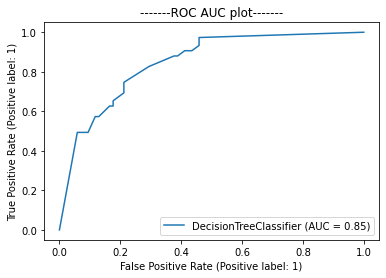


1. **Decision Tree Classifier**

Best parameters: {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 5}

Best Estimator: DecisionTreeClassifier(criterion='entropy', max\_depth=10, min\_samples\_leaf=5)

Final Accuracy with Decision Tree Classifier: 0.76875

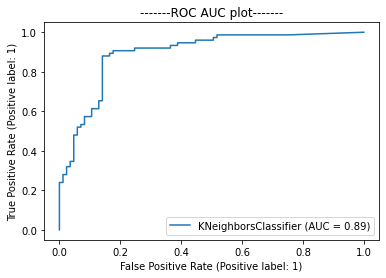


1. **K Neighbors Classifier**

Best parameters: {'metric': 'manhattan', 'n\_neighbors': 9, 'weights': 'distance'}

Best Estimator: KNeighborsClassifier(metric='manhattan', n\_neighbors=9, weights='distance')

Final Accuracy with K Neighbor Classifier: 0.85625

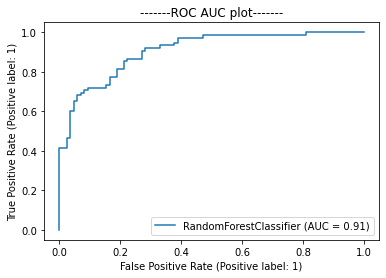


1. **Random Forest Classifier**

Best parameters: {'max\_depth': 10, 'max\_features': 'sqrt', 'min\_samples\_leaf': 1, 'min\_samples\_split': 10, 'n\_estimators': 100}

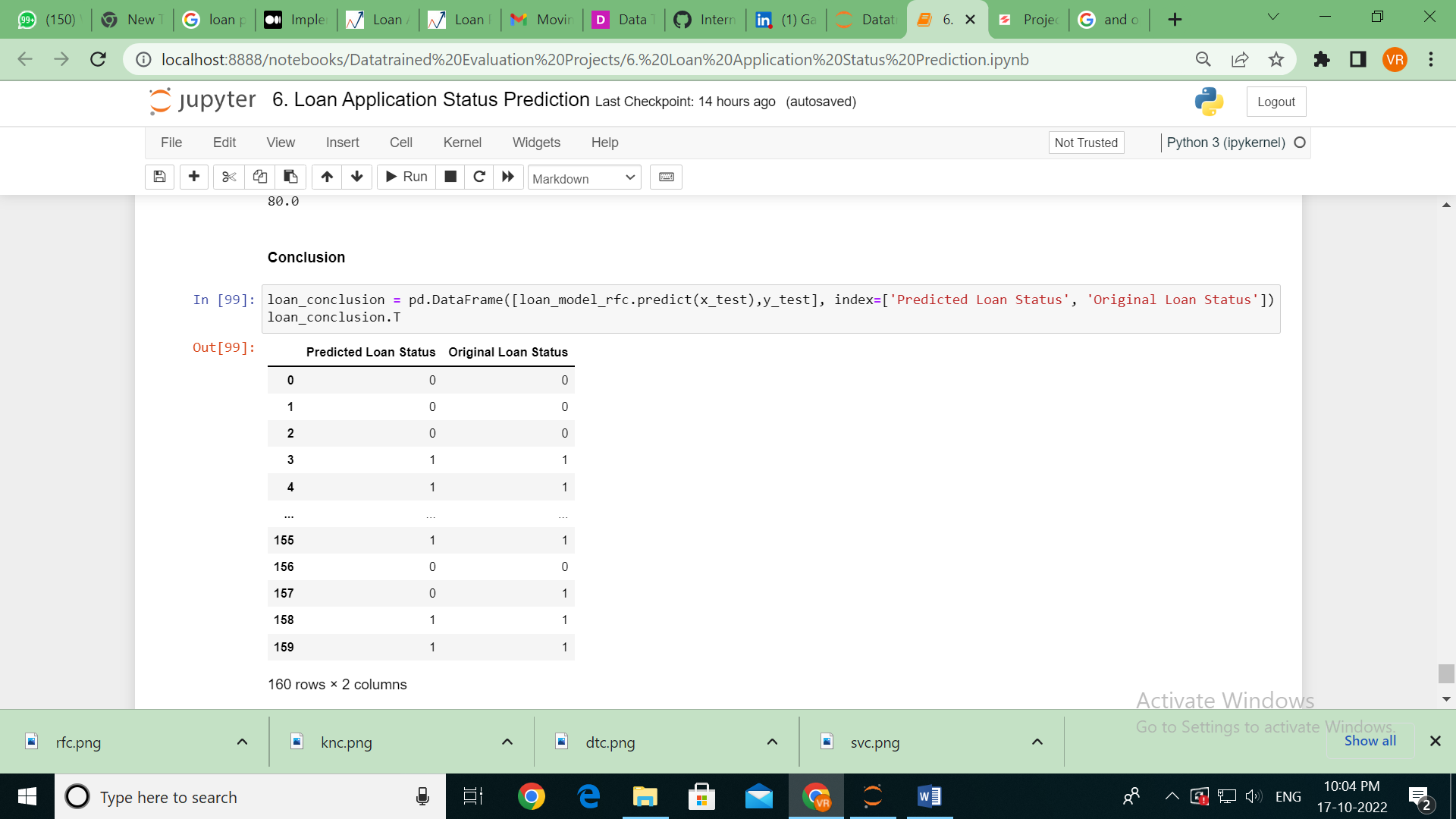
Best Estimator: RandomForestClassifier(max\_depth=10, max\_features='sqrt', min\_samples\_split=10)

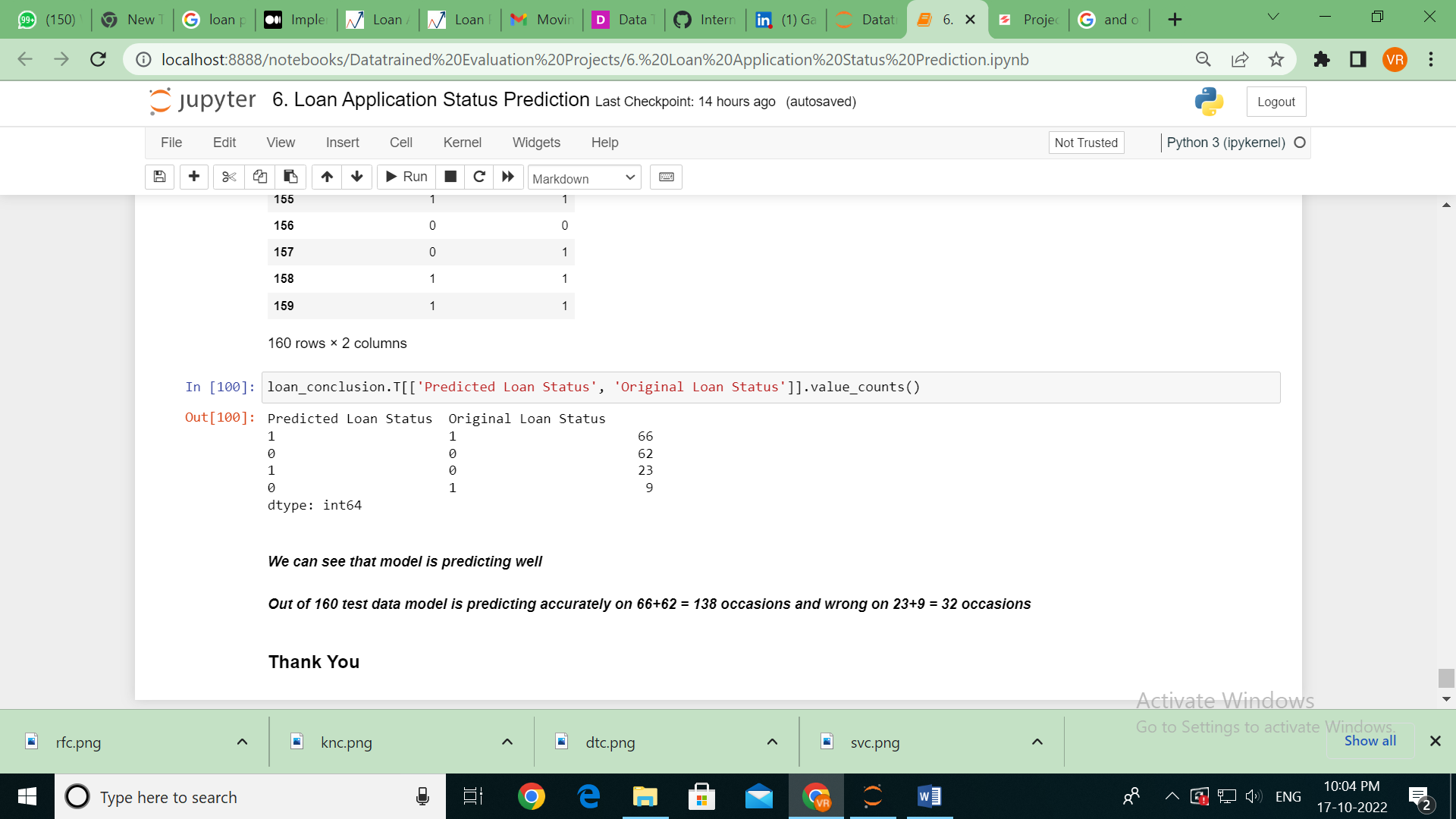
Final Accuracy with Random Forest Classifier: 0.8



* We have best AUC-ROC curve with Random Forest classifier
* Deploying the model with Random forest classifier

**Conclusion**





* We can see that the Random forest classifier model is predicting well.
* Out of 160 test data model is predicting accurately on 66+62 = 138 occasions and wrong on 23+9 = 32 occasions.