IBM NAAN MUDHALVAN

CREDIT CARD FRAUD DETECTION

PHASE 3:

ANALYSIS

# importing data

transaction\_dataset= pd.read\_csv("/content/drive/MyDrive/google\_collab/creditcard.csv")

transaction\_dataset.head(10)

[**Data analysis**](https://github.com/nano-bot01/Credit-Card-Fraud-Detection-/blob/main/README.md#data-analysis)

* shape
* info()
* describe()
* isnull
* count\_values()
* dtypes

**Seperating data for analysis**

* 0 : Normal transaction
* 1 : Fraudulent transaction

legit = transaction\_dataset[transaction\_dataset.Class == 0]

fraud = transaction\_dataset[transaction\_dataset.Class == 1]

print("Shape of legit : ", legit.shape)

print("Shape of fraud : ", fraud.shape)

Shape of legit : (284315, 31)

Shape of fraud : (492, 31)

*# measures*

legit.Amount.describe()

count 284315.000000

mean 88.291022

std 250.105092

min 0.000000

25% 5.650000

50% 22.000000

75% 77.050000

max 25691.160000

Name: Amount, dtype: float64

fraud.Amount.describe()

*# fraud transaction description*

count 492.000000

mean 122.211321

std 256.683288

min 0.000000

25% 1.000000

50% 9.250000

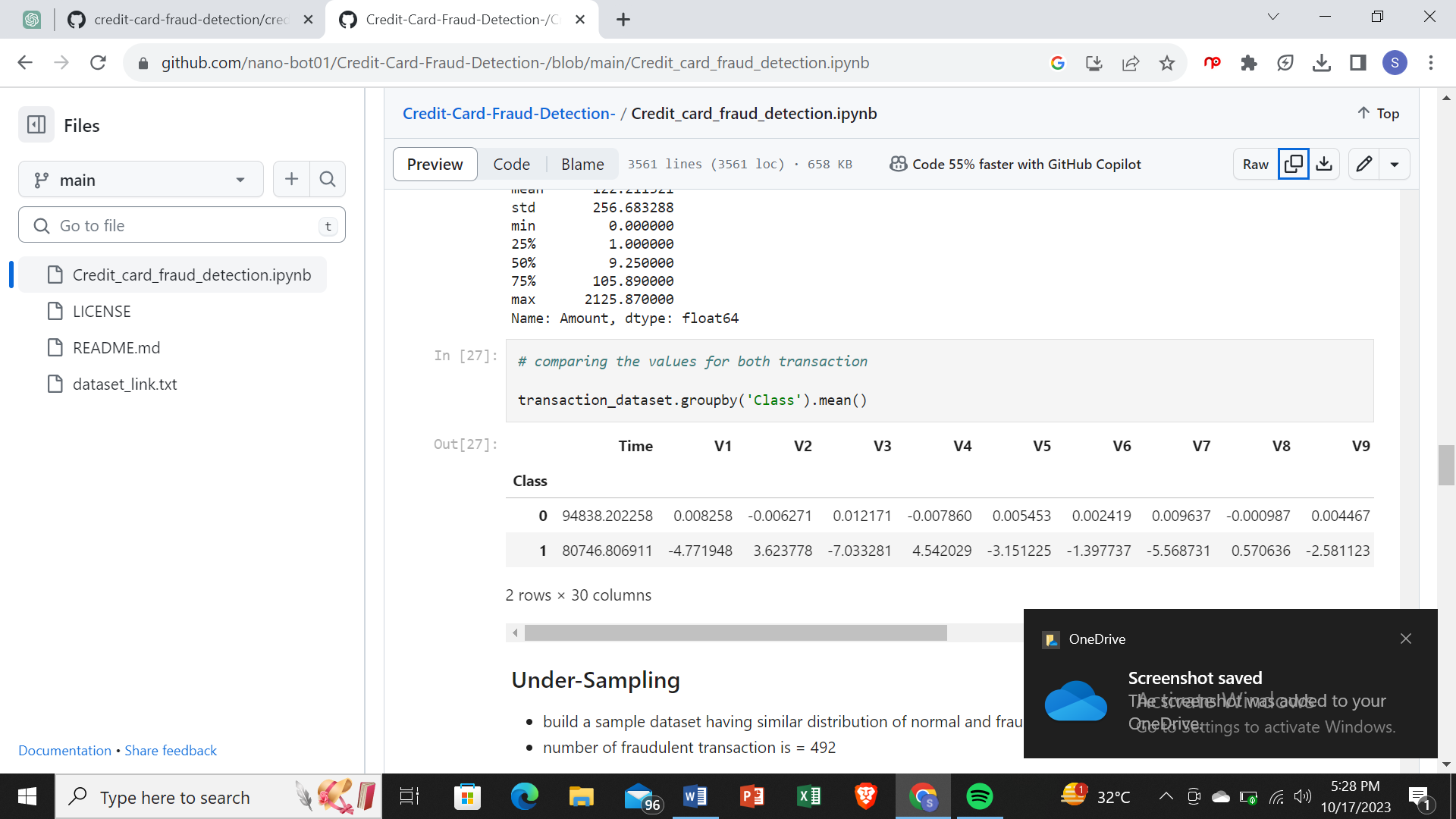
75% 105.890000

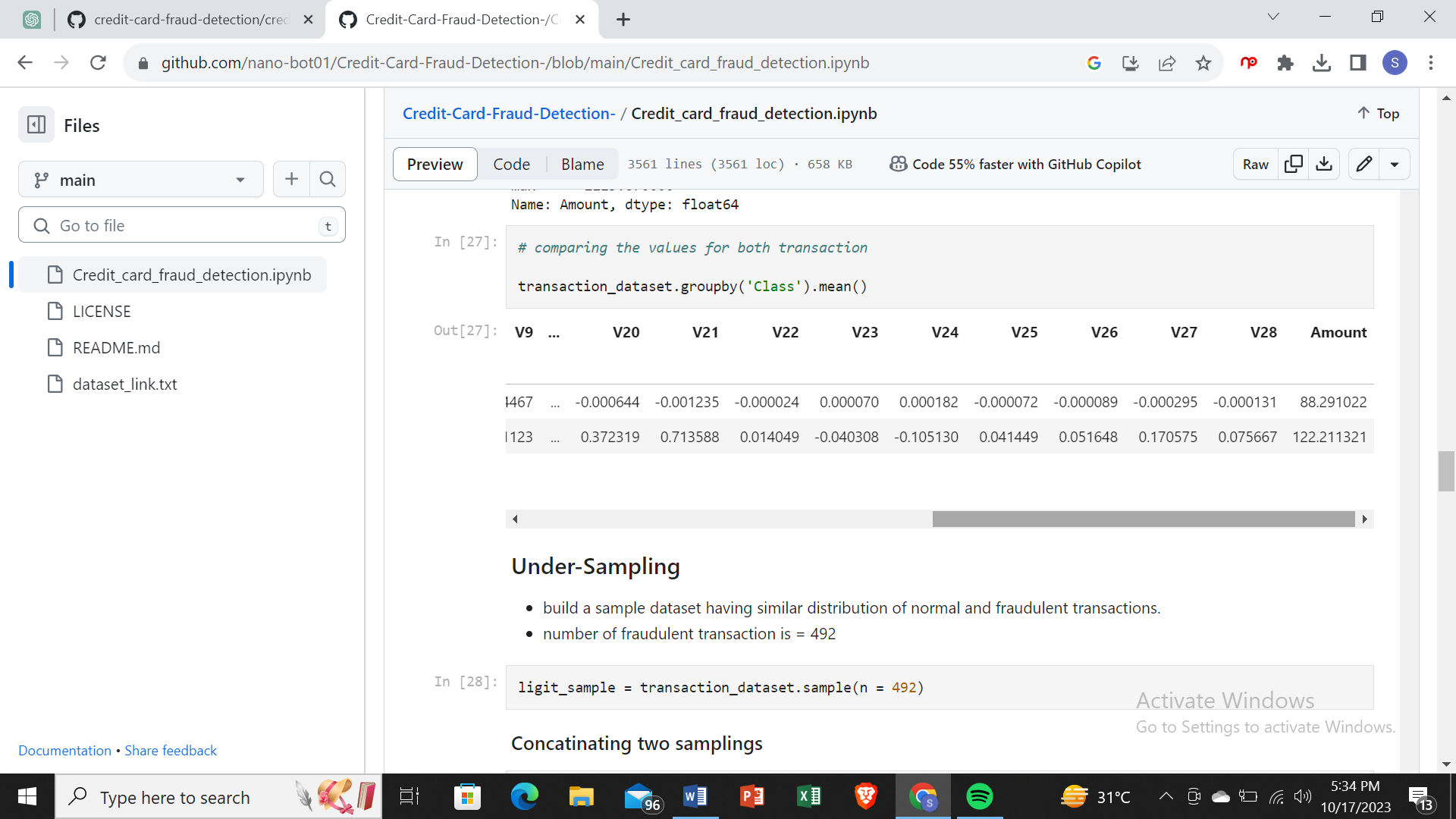
max 2125.870000

Name: Amount, dtype: float64

*# comparing the values for both transaction*

transaction\_dataset.groupby('Class').mean()





**Under-Sampling**

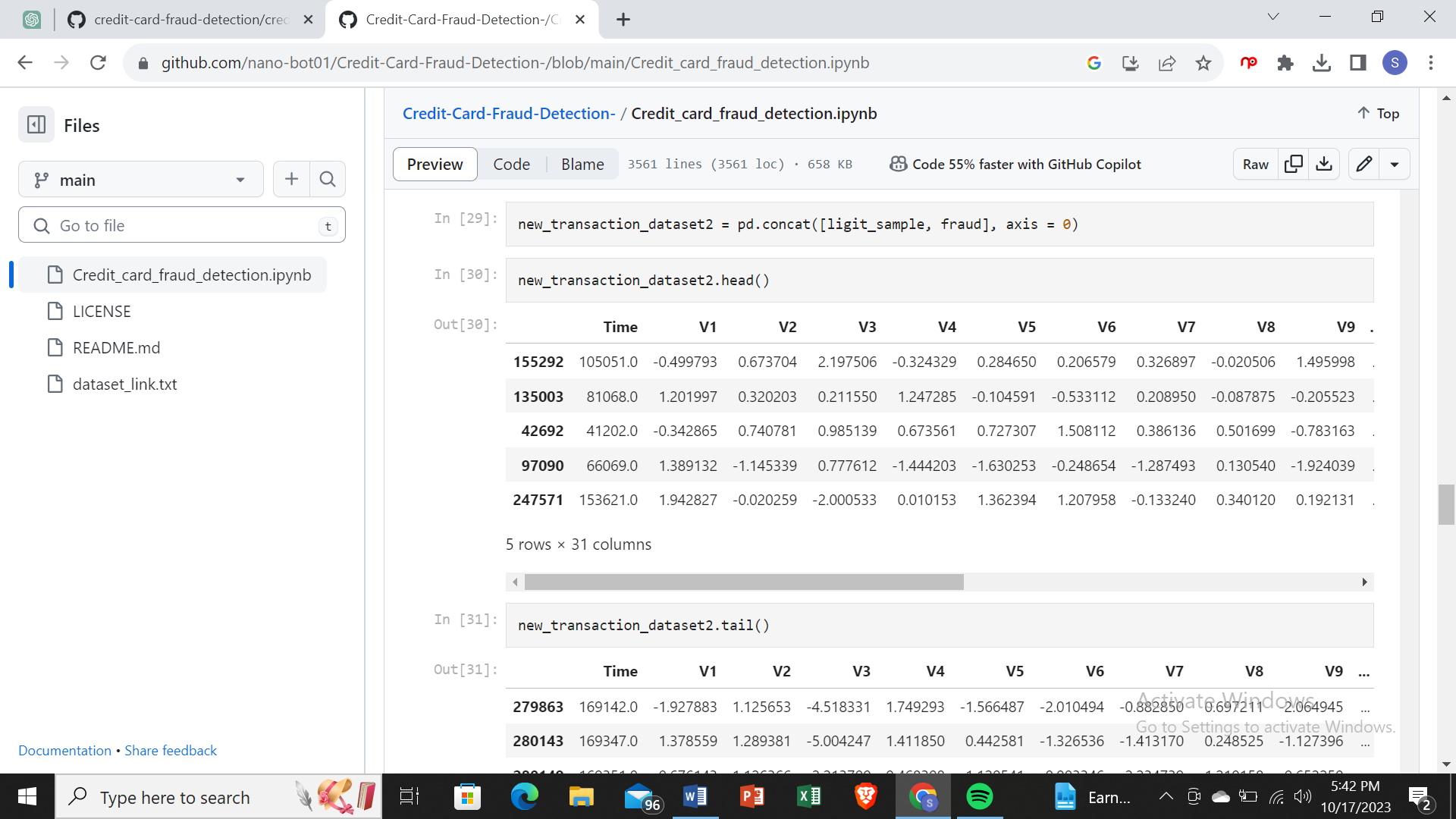
* build a sample dataset having similar distribution of normal and fraudulent transactions.
* number of fraudulent transaction is = 492

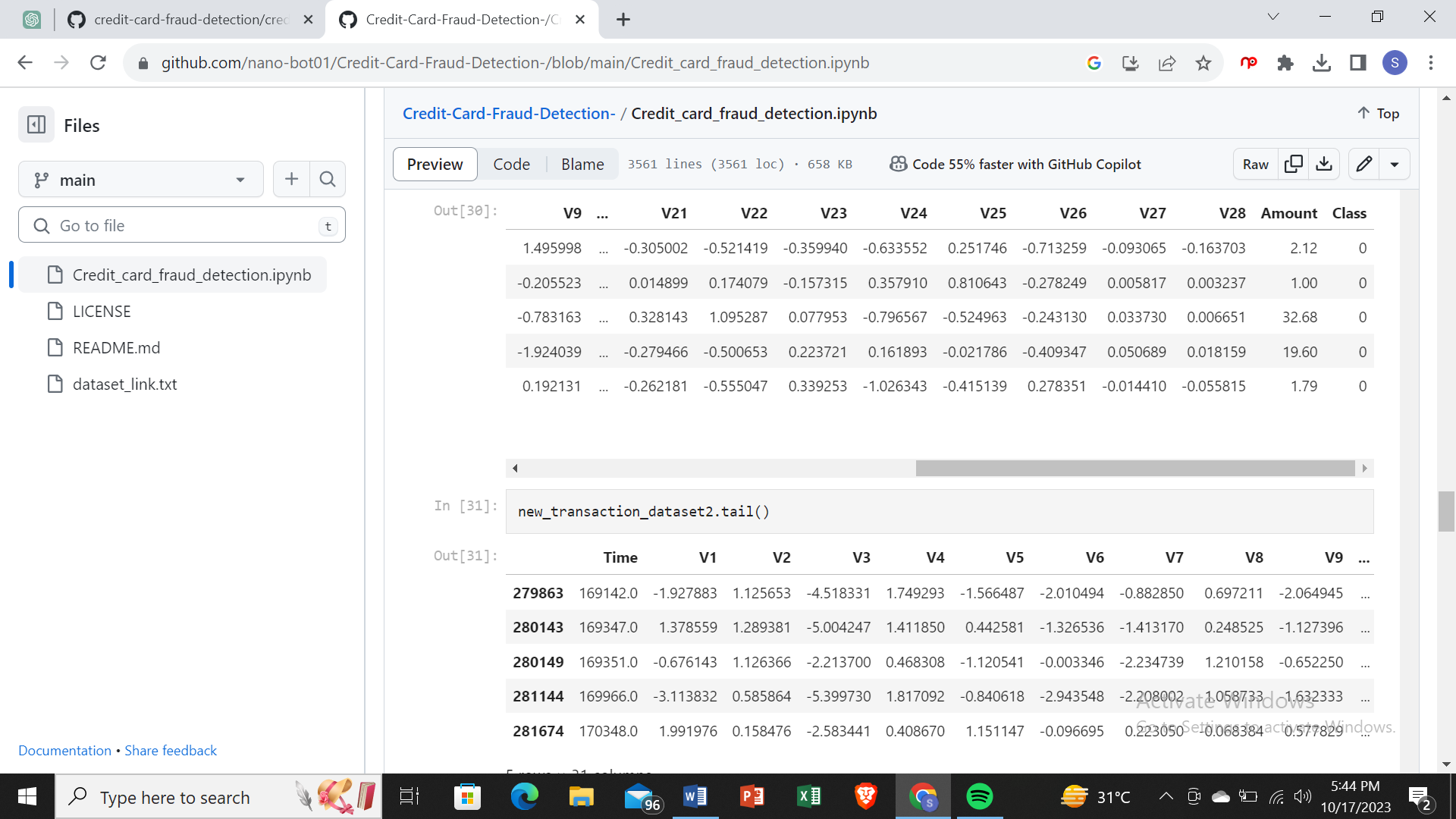
ligit\_sample = transaction\_dataset.sample(n = 492)

**Concatinating two samplings**

new\_transaction\_dataset2 = pd.concat([ligit\_sample, fraud], axis = 0)

new\_transaction\_dataset2.head()





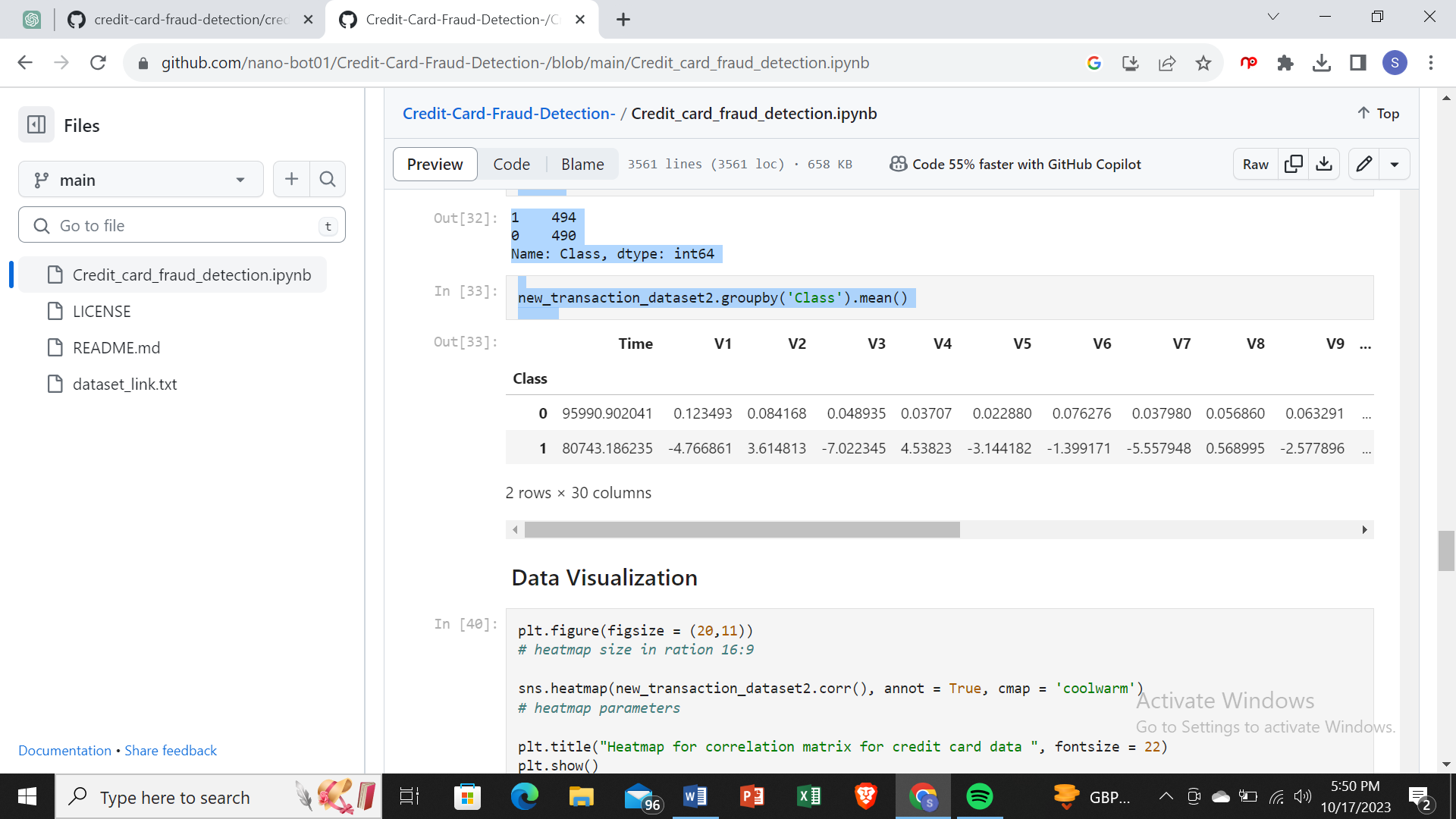
new\_transaction\_dataset2['Class'].value\_counts()

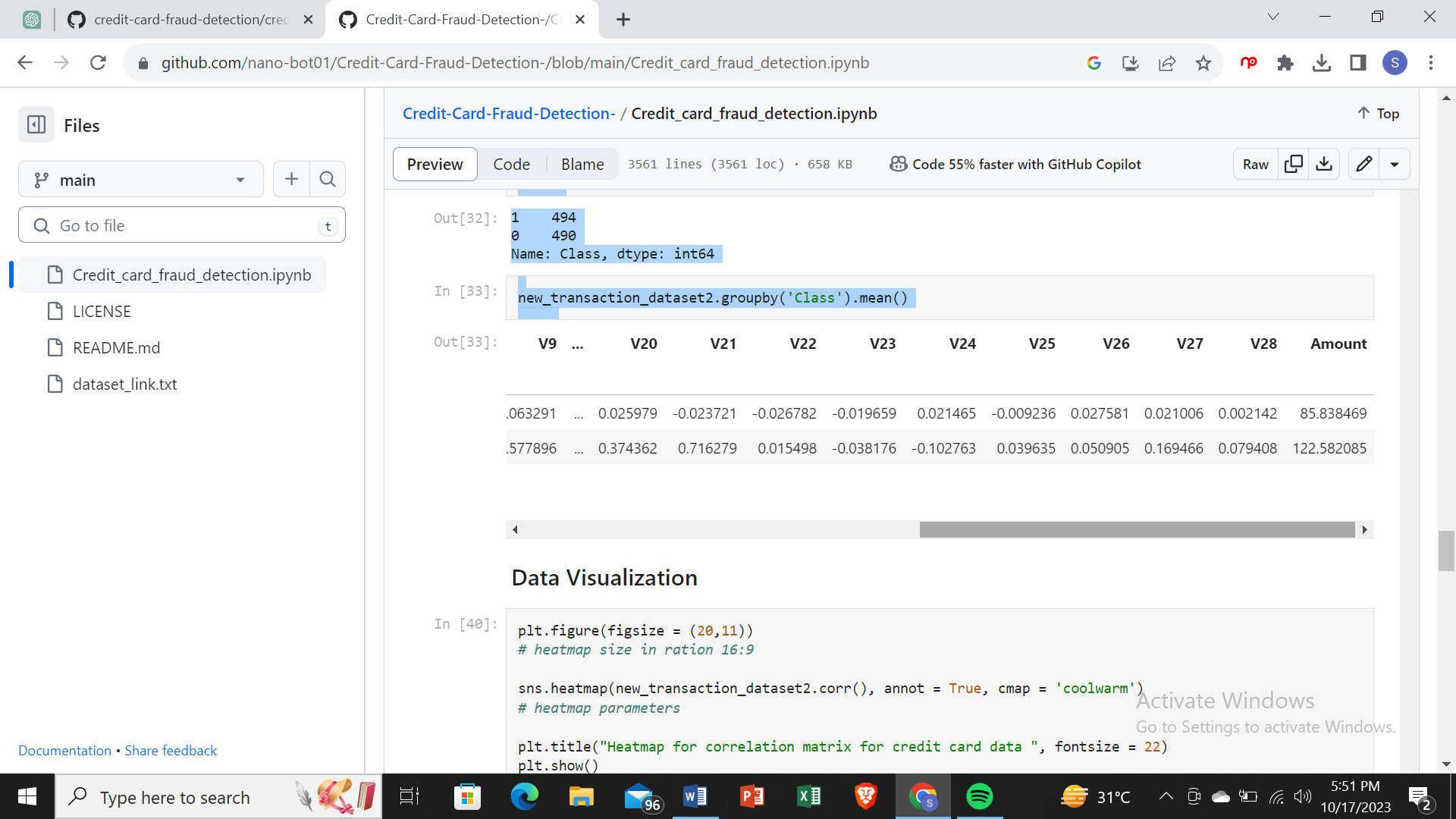
1 494

0 490

Name: Class, dtype: int64

new\_transaction\_dataset2.groupby('Class').mean()





### Data Visualization

plt.figure(figsize = (20,11))

*# heatmap size in ration 16:9*

sns.heatmap(new\_transaction\_dataset2.corr(), annot = True, cmap = 'coolwarm')

*# heatmap parameters*

plt.title("Heatmap for correlation matrix for credit card data ", fontsize = 22)

plt.show()

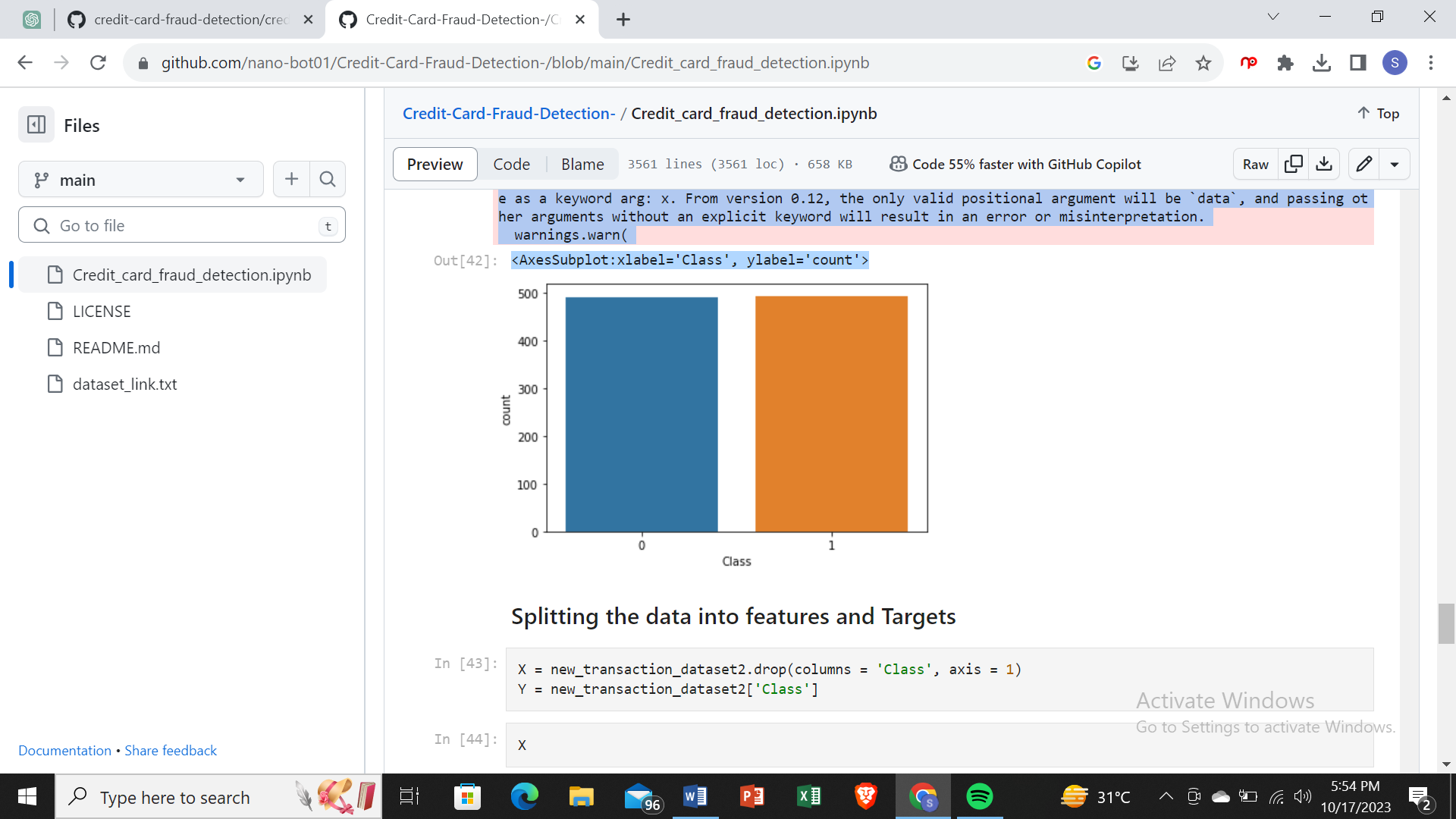


sns.countplot(new\_transaction\_dataset2.Class)

/usr/local/lib/python3.8/dist-packages/seaborn/\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

<AxesSubplot:xlabel='Class', ylabel='count'>

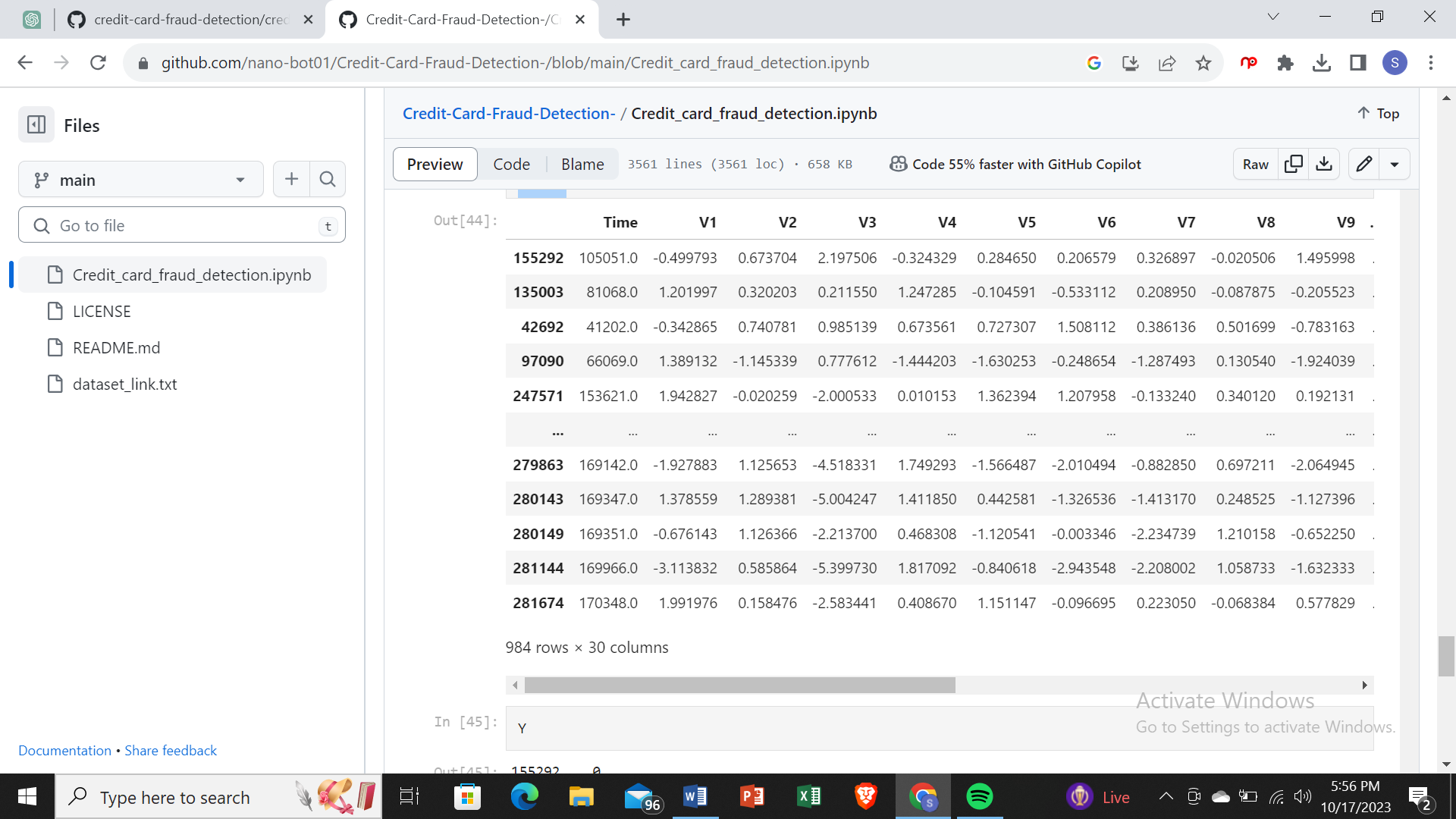


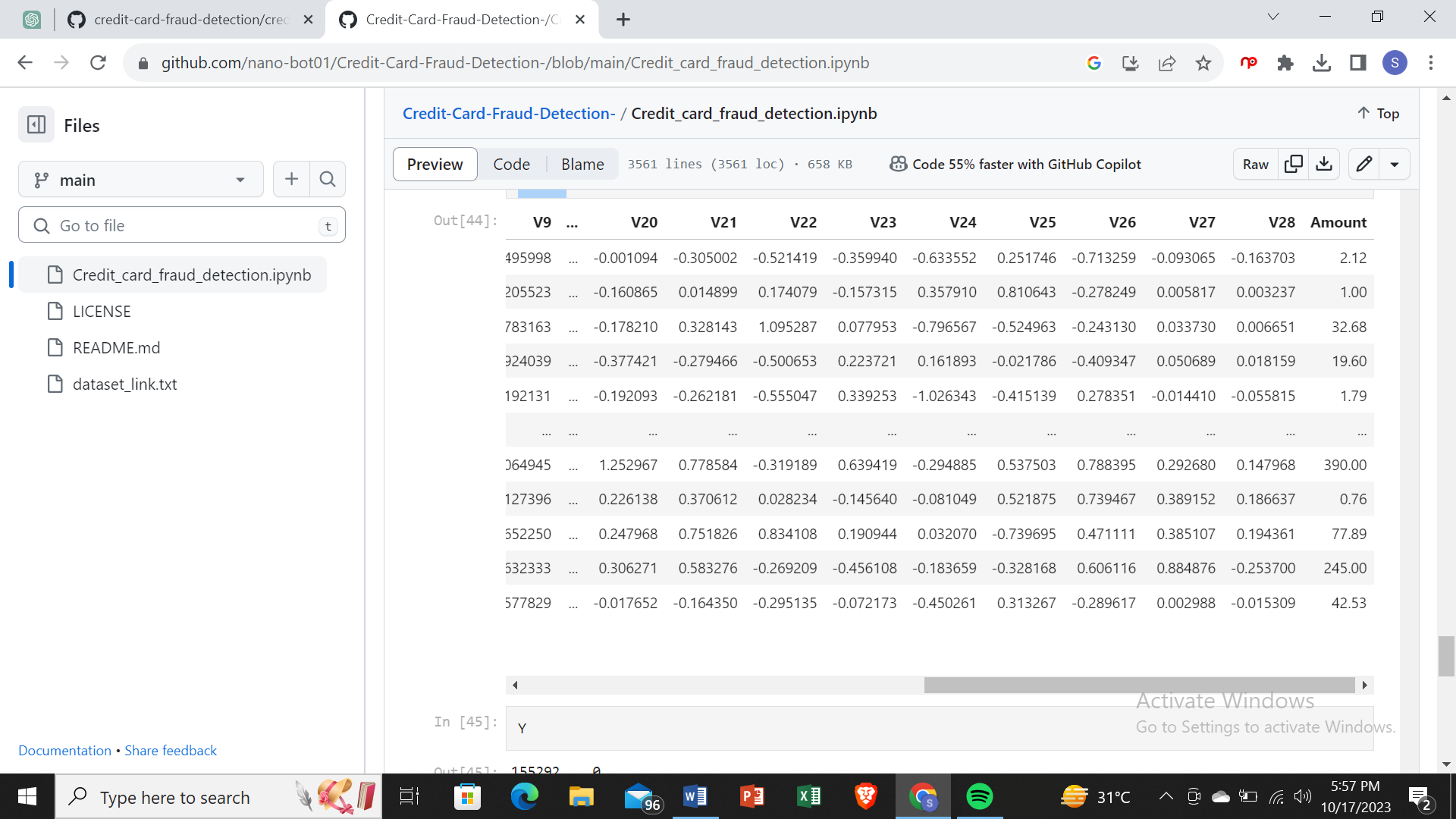
### Splitting the data into features and Targets

X = new\_transaction\_dataset2.drop(columns = 'Class', axis = 1)

Y = new\_transaction\_dataset2['Class']

X





Y

155292 0

135003 0

42692 0

97090 0

247571 0

..

279863 1

280143 1

280149 1

281144 1

281674 1

Name: Class, Length: 984, dtype: int64

### Splitting the data into training and testing data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y, test\_size = 0.2, stratify = Y, random\_state = 2)

print("Shape of X\_train ", X\_train.shape)

print("Shape of X\_test ", X\_test.shape)

print("Shape of Y\_train ", Y\_train.shape)

print("Shape of Y\_test ", Y\_test.shape)

Shape of X\_train (787, 30)

Shape of X\_test (197, 30)

Shape of Y\_train (787,)

Shape of Y\_test (197,)

### Model Training

model = LogisticRegression()

model.fit(X\_train, Y\_train)

LogisticRegression()

## Model Evaluation

* Accuracy Score

*# accuracy on training data*

X\_train\_prediction = model.predict(X\_train)

traning\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print("Accuracy on Training data ",traning\_data\_accuracy)

Accuracy on Training data 0.9390088945362135

*# accuracy on testing data*

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print("Accuracy on Training data ",test\_data\_accuracy)

Accuracy on Training data 0.9441624365482234

Data Preprocessing

SCALING

Standardization and Robust Scalar

Since fraud transactions which are also low in number have relatively smaller value(amount) so we need to have our data scaled, We are going to use robust scalar to scale our data.

Go through sklaern.preprocessing to know about all different kind of scalars present

from sklearn.preprocessing import RobustScaler

rob\_scaler = RobustScaler()

df['scaled\_amount'] = rob\_scaler.fit\_transform(df['Amount'].values.reshape(-1,1))

df.drop(['Time','Amount'],axis=1,inplace=True)

df.drop(['Time\_min','Time\_hour'],axis=1,inplace=True)

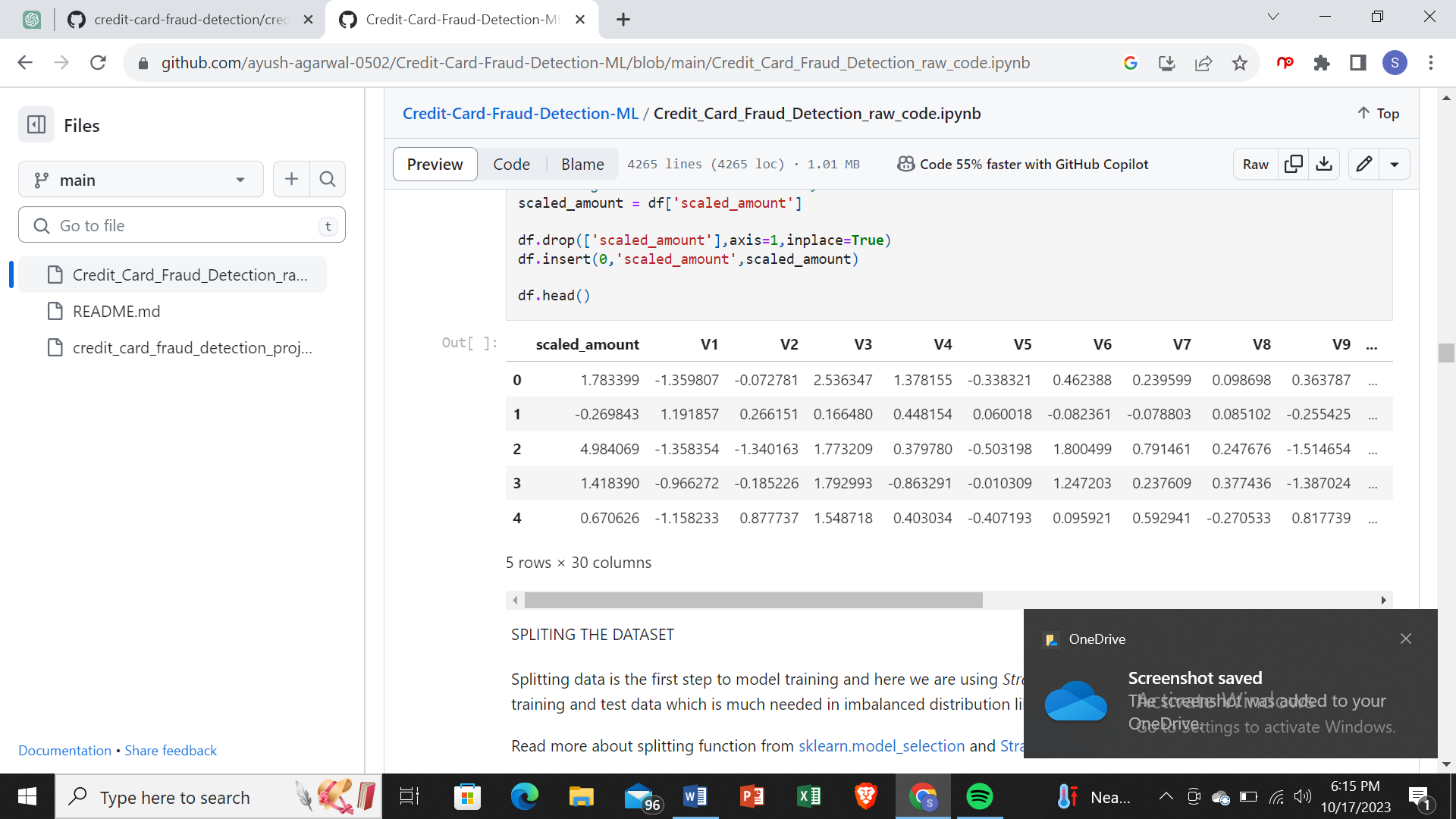
#inserting these scaled columns at 0,1

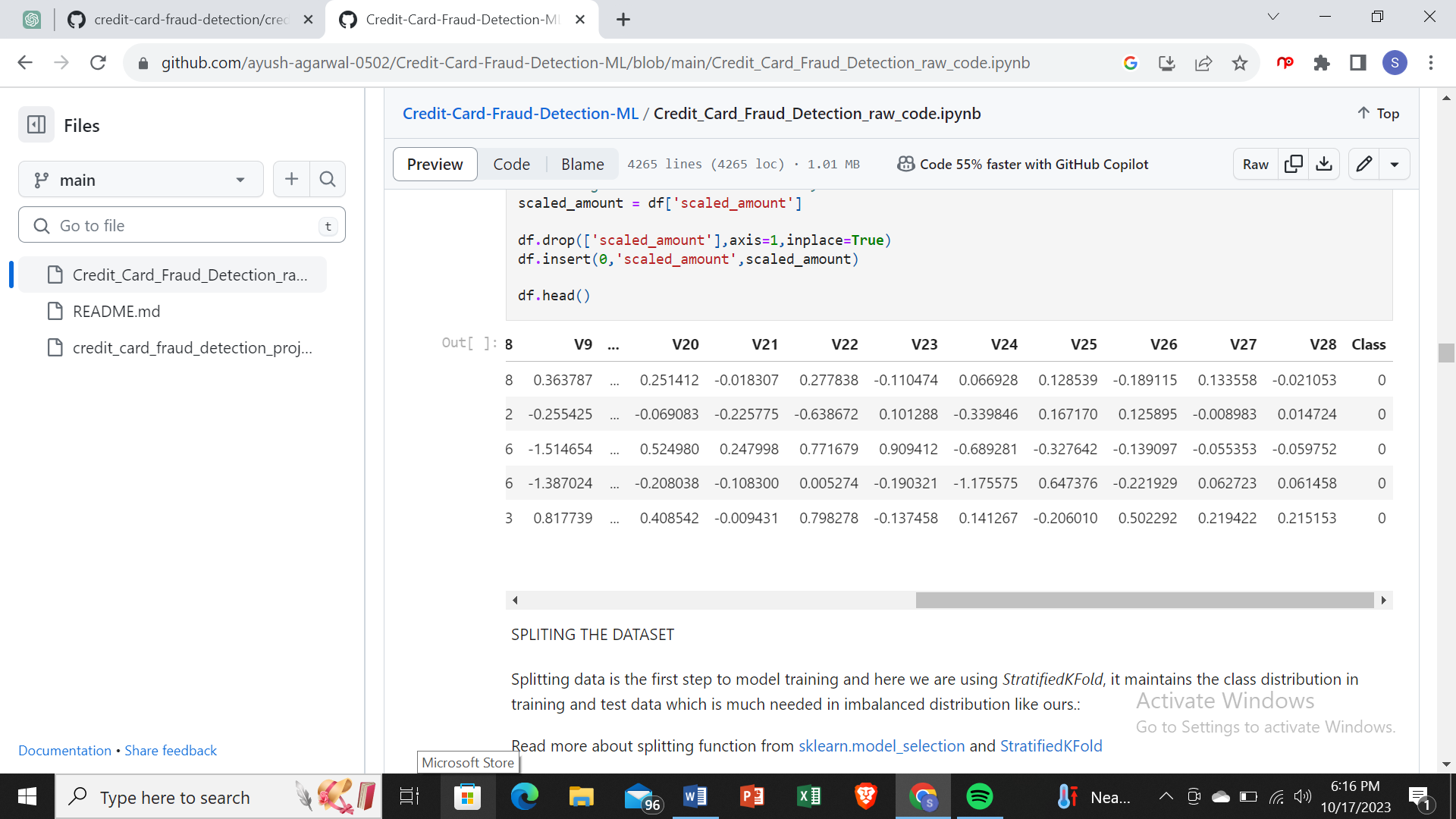
scaled\_amount = df['scaled\_amount']

df.drop(['scaled\_amount'],axis=1,inplace=True)

df.insert(0,'scaled\_amount',scaled\_amount)

df.head()





Splitting The Dataset

Splitting data is the first step to model training and here we are using *StratifiedKFold*, it maintains the class distribution in training and test data which is much needed in imbalanced distribution like ours.:

Read more about splitting function from [sklearn.model\_selection](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection) and [StratifiedKFold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold)

fr**om** sklearn.model\_selection **import** StratifiedKFold

X **=** df**.**drop('Class',axis**=**1)

y **=** df['Class']

sss **=** StratifiedKFold(n\_splits**=**5,random\_state**=None**,shuffle**=False**)

**for** train\_index, test\_index **in** sss**.**split(X,y):

print("Train:", train\_index, "Test:", test\_index)

original\_Xtrain, original\_Xtest **=** X**.**iloc[train\_index], X**.**iloc[test\_index]

original\_ytrain, original\_ytest **=** y**.**iloc[train\_index],y**.**iloc[test\_index]

*#converting it into an array*

original\_Xtrain **=** original\_Xtrain**.**values

original\_Xtest **=** original\_Xtest**.**values

original\_ytrain **=** original\_ytrain**.**values

original\_ytest **=** original\_ytest**.**values

*#check if both train and test distributions are similarly distributed*

train\_unique\_label, train\_counts\_label **=** np**.**unique(original\_ytrain, return\_counts**=True**)

test\_unique\_label, test\_counts\_label **=** np**.**unique(original\_ytest,return\_counts**=True**)

print("Label dstributions: \n")

print(train\_counts\_label**/**len(original\_ytrain))

print(test\_counts\_label**/**len(original\_ytest))

Train: [ 30442 30473 30496 ... 284796 284797 284798] Test: [ 0 1 2 ... 57016 57017 57018]

Train: [ 0 1 2 ... 284796 284797 284798] Test: [ 30442 30473 30496 ... 113962 113963 113964]

Train: [ 0 1 2 ... 284796 284797 284798] Test: [ 80757 81183 81606 ... 170942 170943 170944]

Train: [ 0 1 2 ... 284796 284797 284798] Test: [150644 150651 150657 ... 227860 227861 227862]

Train: [ 0 1 2 ... 227860 227861 227862] Test: [212511 212639 213087 ... 284796 284797 284798]

Label dstributions:

[0.99827072 0.00172928]

[0.99827946 0.00172054]

UNDERSAMPLING TO MAKE THE DATASET BALANCED

since our classes are highly skewed, we have to make them equivalent in occurence to have a normal distribution of the classes, shuffle the data before creating the sub-samples.

df **=** df**.**sample(frac**=**1)

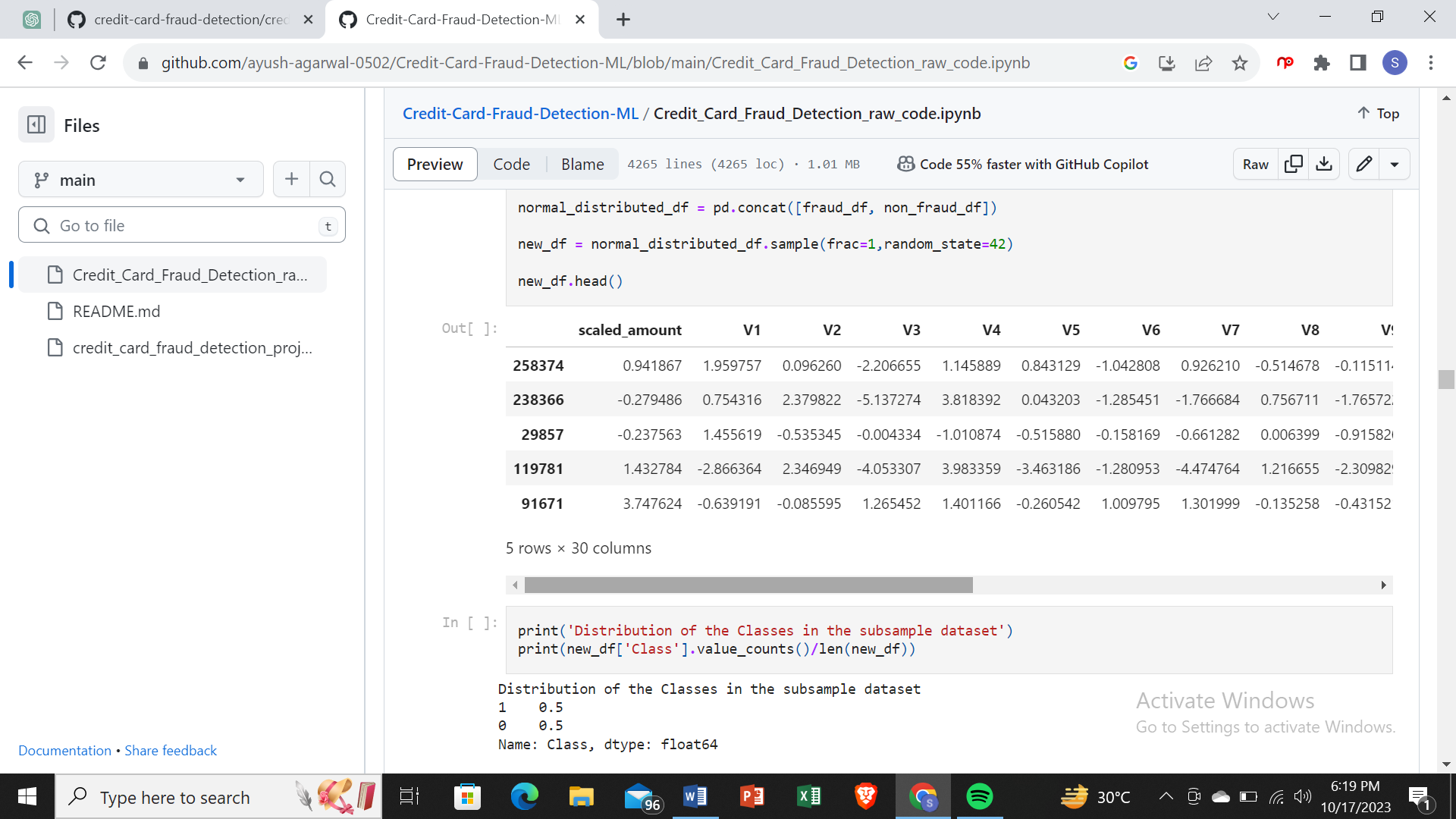
fraud\_df **=** df**.**loc[df['Class']**==**1]

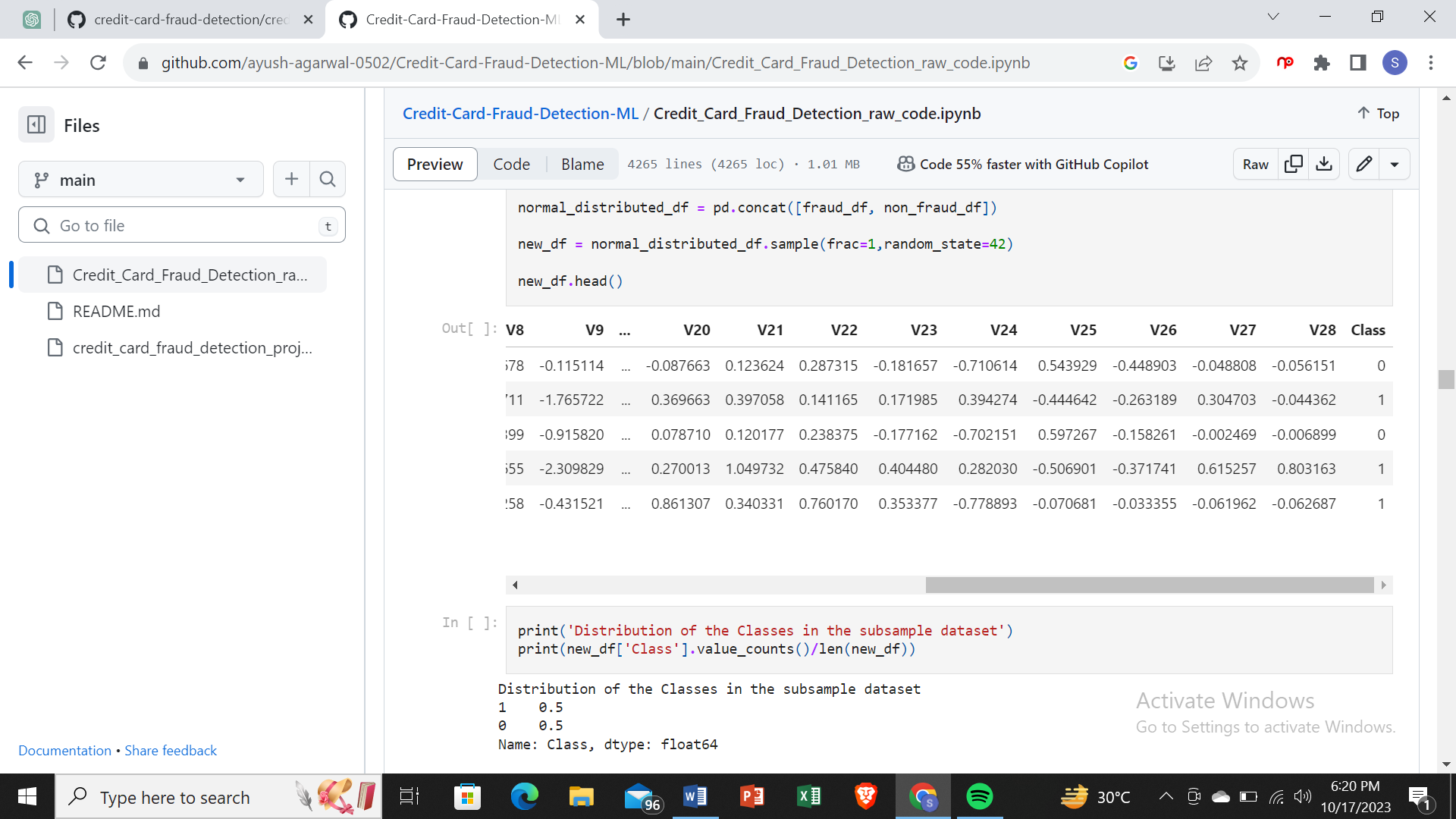
non\_fraud\_df **=** df**.**loc[df['Class']**==**0][:492]

normal\_distributed\_df **=** pd**.**concat([fraud\_df, non\_fraud\_df])

new\_df **=** normal\_distributed\_df**.**sample(frac**=**1,random\_state**=**42)

new\_df**.**head()





print('Distribution of the Classes in the subsample dataset')

print(new\_df['Class']**.**value\_counts()**/**len(new\_df))

Distribution of the Classes in the subsample dataset

1 0.5

0 0.5

Name: Class, dtype: float64

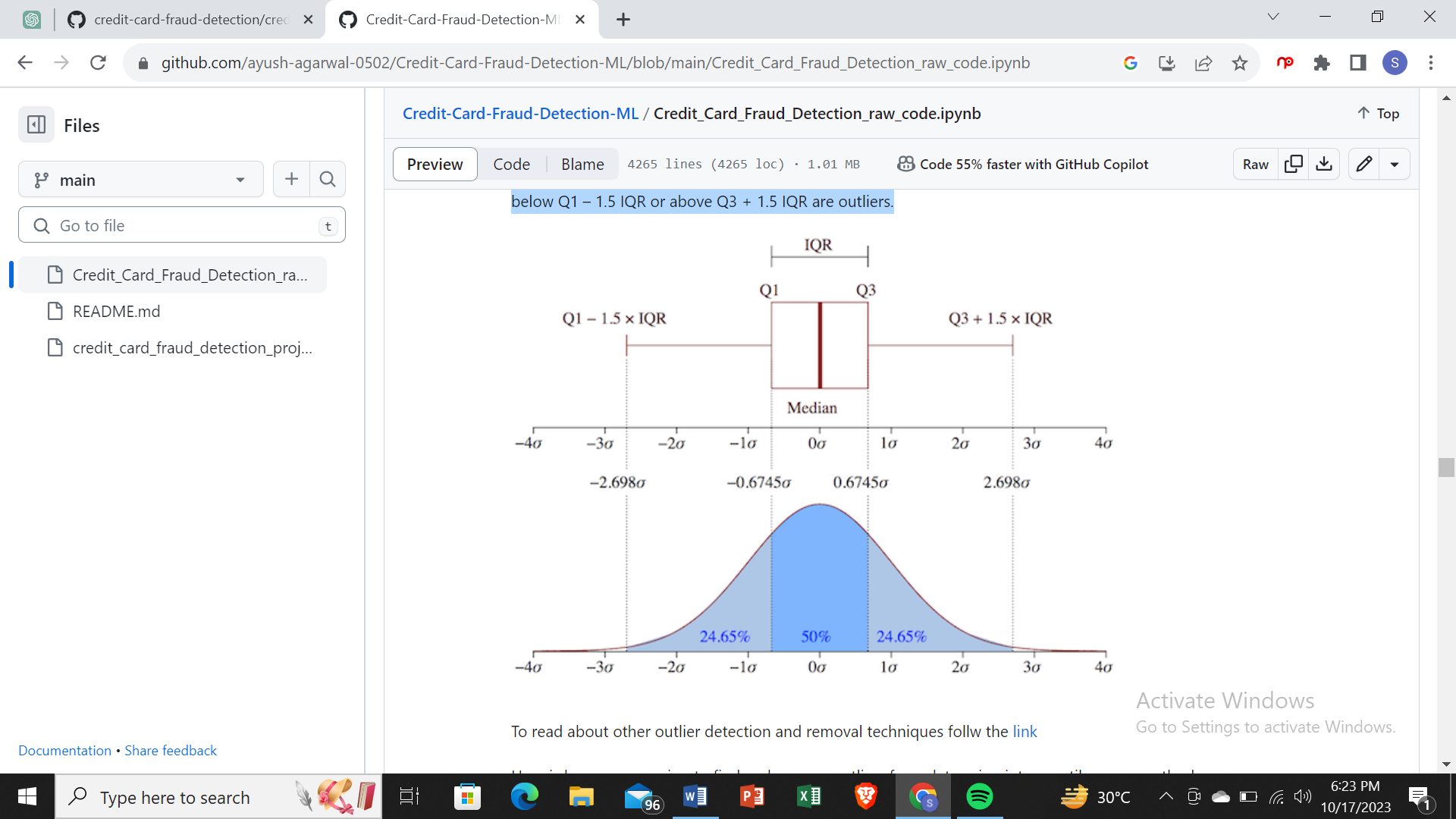
REMOVING OUTLIERS

We will use interquatile range to remove outliers from highly correlated features

IQR is used to measure variability by dividing a data set into quartiles. The data is sorted in ascending order and split into 4 equal parts. Q1, Q2, Q3 called first, second and third quartiles are the values which separate the 4 equal parts.

* Q1 represents the 25th percentile of the data.
* Q2 represents the 50th percentile of the data.
* Q3 represents the 75th percentile of the data.

IQR is the range between the first and the third quartiles namely Q1 and Q3: IQR = Q3 – Q1. The data points which fall below Q1 – 1.5 IQR or above Q3 + 1.5 IQR are outliers.



To read about other outlier detection and removal techniques follw the [link](https://https/towardsdatascience.com/5-ways-to-detect-outliers-that-every-data-scientist-should-know-python-code-70a54335a623)

Here is how we are going to find and remove outliers from data using interquartile range method

Go through the example to complete the cell below and run for yourself

v14\_fraud = new\_df['V14'].loc[new\_df['Class'] == 1].values

q25, q75 = np.percentile(v14\_fraud, 25), np.percentile(v14\_fraud, 75)

v14\_iqr = q75 - q25

v14\_cut\_off = v14\_iqr \* 1.5

v14\_lower, v14\_upper = q25 - v14\_cut\_off, q75 + v14\_cut\_off

outliers = [x for x in v14\_fraud if x < v14\_lower or x > v14\_upper]

new\_df = new\_df.drop(new\_df[(new\_df['V14'] > v14\_upper) | (new\_df['V14'] < v14\_lower)].index)

*#Complete Line 2-16*

v14\_fraud **=** new\_df['V14']**.**loc[new\_df['Class'] **==** 1]**.**values

q25, q75 **=**np**.**percentile(v14\_fraud, 25), np**.**percentile(v14\_fraud, 75)

print('Quartile 25: {} | Quartile 75: {}'**.**format(q25, q75))

v14\_iqr **=**q75 **-** q25

print('iqr: {}'**.**format(v14\_iqr))

v14\_cut\_off **=**v14\_iqr **\*** 1.5

v14\_lower, v14\_upper **=** q25 **-** v14\_cut\_off, q75 **+** v14\_cut\_off

print('Cut Off: {}'**.**format(v14\_cut\_off))

print('V14 Lower: {}'**.**format(v14\_lower))

print('V14 Upper: {}'**.**format(v14\_upper))

outliers **=** [x **for** x **in** v14\_fraud **if** x **<** v14\_lower **or** x **>** v14\_upper]

print('Feature V14 Outliers for Fraud Cases: {}'**.**format(len(outliers)))

print('V14 outliers:{}'**.**format(outliers))

new\_df **=** new\_df**.**drop(new\_df[(new\_df['V14'] **>** v14\_upper) **|** (new\_df['V14'] **<** v14\_lower)]**.**index)

print('----' **\*** 44)

*###################################################################################################################*

v12\_fraud **=** new\_df['V12']**.**loc[new\_df['Class'] **==** 1]**.**values

q25, q75 **=** np**.**percentile(v12\_fraud, 25), np**.**percentile(v12\_fraud, 75)

v12\_iqr **=** q75 **-** q25

v12\_cut\_off **=** v12\_iqr **\*** 1.5

v12\_lower, v12\_upper **=** q25 **-** v12\_cut\_off, q75 **+** v12\_cut\_off

print('V12 Lower: {}'**.**format(v12\_lower))

print('V12 Upper: {}'**.**format(v12\_upper))

outliers **=** [x **for** x **in** v12\_fraud **if** x **<** v12\_lower **or** x **>** v12\_upper]

print('V12 outliers: {}'**.**format(outliers))

print('Feature V12 Outliers for Fraud Cases: {}'**.**format(len(outliers)))

new\_df **=** new\_df**.**drop(new\_df[(new\_df['V12'] **>** v12\_upper) **|** (new\_df['V12'] **<** v12\_lower)]**.**index)

print('Number of Instances after outliers removal: {}'**.**format(len(new\_df)))

print('----' **\*** 44)

*##################################################################################################################*

v10\_fraud **=** new\_df['V10']**.**loc[new\_df['Class'] **==** 1]**.**values

q25, q75 **=** np**.**percentile(v10\_fraud, 25), np**.**percentile(v10\_fraud, 75)

v10\_iqr **=** q75 **-** q25

v10\_cut\_off **=** v10\_iqr **\*** 1.5

v10\_lower, v10\_upper **=** q25 **-** v10\_cut\_off, q75 **+** v10\_cut\_off

print('V10 Lower: {}'**.**format(v10\_lower))

print('V10 Upper: {}'**.**format(v10\_upper))

outliers **=** [x **for** x **in** v10\_fraud **if** x **<** v10\_lower **or** x **>** v10\_upper]

print('V10 outliers: {}'**.**format(outliers))

print('Feature V10 Outliers for Fraud Cases: {}'**.**format(len(outliers)))

new\_df **=** new\_df**.**drop(new\_df[(new\_df['V10'] **>** v10\_upper) **|** (new\_df['V10'] **<** v10\_lower)]**.**index)

print('Number of Instances after outliers removal: {}'**.**format(len(new\_df)))

Quartile 25: -9.692722964972385 | Quartile 75: -4.282820849486865

iqr: 5.40990211548552

Cut Off: 8.11485317322828

V14 Lower: -17.807576138200666

V14 Upper: 3.832032323741415

Feature V14 Outliers for Fraud Cases: 4

V14 outliers:[-18.049997689859396, -18.4937733551053, -18.8220867423816, -19.2143254902614]

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

V12 Lower: -17.3430371579634

V12 Upper: 5.776973384895937

V12 outliers: [-18.683714633344298, -18.4311310279993, -18.553697009645802, -18.047596570821604]

Feature V12 Outliers for Fraud Cases: 4

Number of Instances after outliers removal: 976

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

V10 Lower: -14.89885463232024

V10 Upper: 4.920334958342141

V10 outliers: [-22.1870885620007, -15.563791338730098, -23.2282548357516, -14.9246547735487, -15.2318333653018, -16.2556117491401, -16.7460441053944, -24.5882624372475, -18.9132433348732, -16.6011969664137, -14.9246547735487, -18.2711681738888, -17.141513641289198, -22.1870885620007, -24.403184969972802, -15.563791338730098, -15.2399619587112, -15.124162814494698, -15.2399619587112, -15.1237521803455, -19.836148851696, -15.346098846877501, -16.3035376590131, -22.1870885620007, -20.949191554361104, -16.6496281595399, -22.1870885620007]

Feature V10 Outliers for Fraud Cases: 27

Number of Instances after outliers removal: 945

f, (ax1,ax2,ax3) **=** plt**.**subplots(1,3, figsize**=**(20,6))

colors **=** ['red','green']

*#feature V14*

sns**.**boxplot(x**=**'Class',y**=**'V14',data**=**new\_df,ax**=**ax1,palette**=**colors)

ax1**.**set\_title("V14 Feature \n Reduction of outliers", fontsize**=**14)

ax1**.**annotate('Fewer extreme \n outliers', xy**=**(0.98, **-**17.5), xytext**=**(0, **-**12),arrowprops**=**dict(facecolor**=**'black'),fontsize**=**14)

*#feature v12*

sns**.**boxplot(x**=**'Class',y**=**'V12',data**=**new\_df,ax**=**ax2,palette**=**colors)

ax2**.**set\_title("V12 Feature \n Reduction of outliers", fontsize**=**14)

ax2**.**annotate('Fewer extreme \n outliers', xy**=**(0.98, **-**17.3), xytext**=**(0, **-**12),arrowprops**=**dict(facecolor**=**'black'),fontsize**=**14)

*#feature V10*

sns**.**boxplot(x**=**'Class',y**=**'V10',data**=**new\_df,ax**=**ax3,palette**=**colors)

ax3**.**set\_title("V10 Feature \n Reduction of outliers", fontsize**=**14)

ax3**.**annotate('Fewer extreme \n outliers', xy**=**(0.95, **-**16.5), xytext**=**(0, **-**12),arrowprops**=**dict(facecolor**=**'black'),fontsize**=**14)

Text(0, -12, 'Fewer extreme \n outliers')

