CREDIT CARD FRAUD DETECTION

A PROJECT REPORT

for

SOFT COMPUTING(SWE1011)- C2+TC2

in

M.TECH SOFTWARE ENGINEERING

by

18MIS0019 – J. SANJAY
18MIS0044 – R. YESWANTH

WINTER SEMESTER, 2021

Under the Guidance of

Prof. SUBHASHINI.R

Associate Professor, SITE

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

1. ABSTRACT:

Nowadays online transactions have grown in large quantities. Among them, online credit card transactions hold a huge share. Therefore, there is much need for credit card fraud detection applications in bans and financial business. Credit card fraud purposes may be to obtain goods without paying or to obtain unauthorized funds from an account. With the demand for money credit card fraud events became common. This results in a huge loss in finances to the cardholder. Previously they used the most common methods like rule induction techniques, fuzzy system, decision trees, Support Vector Machines (SVM), K-Nearest Neighbor algorithms to detect the fraud transaction using a credit card. From our perspective, neural networks will generate more accurate results.

To increase the accuracy and precision we use the algorithms Logistic Regression, Convolution Neural Networks. Logistic Regression is a statistical model that tries to minimize the cost of how wrong a prediction is. CNN algorithm is used, to capture the intrinsic patterns of fraud behaviors learned from labeled data. So will make use of accuracy and precision to evaluate the performance of the proposed system

2. INTRODUCTION:

A credit card fraud can be done in the subsequent ways: 1. Fake IDs: This type of fraud is done by using the personal information of other persons without any authorization. 2. Skimming Method: This type of fraud is done by installing a device called “skimmer” at ATM machines. The data present on the magnetic stripe of the card is collected when the card is swiped 3. Card not present: When the fraudster steals the data like the expiry date and account number of the card, they can use the card without being possessed physically. 4. False businessperson sites: this type of fraud is done by the method Phishing, where they create a website/webpage which looks similar to that of the original site.

Convolution Neural Networks algorithms are used to achieve high accuracy rates and to increase the detection process. Convolution Neural networks are used to identify the underlying patterns that are followed in the previous cases and thereby increasing efficiency.

3. LITERATURE SURVEY:

3.1 A Cost-Sensitive Decision Tree Approach for Fraud Detection:

As the information technology is developing the fraud is also increasing as a result financial loss due to fraud is also very large. A cost sensitive decision tree approach has been used for fraud detection. A cost called misclassification cost is used which is taken as varying as well as priorities of the fraud also differs according to individual records. So common performance metrics such as accuracy, True Positive Rate (TPR) or even area Under Curve cannot be used to evaluate the performance of the models because they accept each fraud as having the same priority regardless of the amount of that fraudulent transaction or the available usable limit of the card used in the transaction at that time. Different methods are used for cost sensitivity. They mainly include the machine learning approach, decision tree approach. In machine learning approach two techniques called over sampling and under sampling is performed, in which the latter obtained a good result

3.2 Credit Card Fraud Detection Techniques:

Data and Technique Oriented Perspective In this study mainly two approaches namely misuses (supervised) and anomaly detection (unsupervised) technique is being used. After this a classification is also used for checking the capability to process categorical and numerical data. In the first approach the data is classified as fraud based on previous data. With the help of this dataset classification models are also created, which can predict whether the data is fraud or not. The different classification models used are decision tree, neural network, rule induction etc. This has obtained a successful result and this approach is also called as misuse approach. While the second approach is based on account behavior. A transaction is said to be fraudulent if it possess the features opposite to the user’s normal behavior.

3.3 Credit Card Fraud Detection Using Hidden Markov Model

As the E-commerce technology is increasing day by day the use of credit card has also been increased. As a result of this the fraud using credit card is also increasing. In all fraud detection systems, fraud will be detected only after the fraud has taken place. In this study a sequence of operations are modelled using Hidden Markov Model (HMM) and this can be used for the detection of fraud. It is trained with the normal behavior of the card holder. If the incoming transaction is not accepted by the trained HMM with high probability it is considered as fraudulent otherwise not.

The main advantage is that it does not require fraud signatures it is capable to detect by bearing in mind card holders spending habit. This is done by creating a set of clusters and identify the spending profile. These data are stored in the form of clusters with low, medium and high values. The probability is based on the spending behavior and further the processing is done

3.4 Real Time Credit Card Fraud Detection using Computational Intelligence

As the growth of technology is increasing it has made a big impact in credit transactions. This has made the fraudsters to commit the fraud transactions easily. Many fraud detection techniques are available but cannot solve fraud problems easily. As a solution to all this, SOM (Self Organizing Map) is used. It helps to decipher, filter and analyze customer behavior for fraud detection. SOM is a unsupervised neural network algorithm which is used to configure neurons according to the topological structure of input data. It divides the data into genuine and fraudulent transactions sets. This system not only deals with customer profiles, merchant profiles and their selling price. It should also include rules and policies in the market place. By involving these attributes, it increases the accuracy to detect the fraud. SOM helps to detect fraud to a great extent

3.5 A Novel Machine Learning Algorithm to credit card fraud detection

The use of credit card is rising day by day as the e-commerce is also increasing. The problem that happens with this is that fraud using credit card is increasing. It is a recurrent problem in almost all countries. But the trend seen is that countries with more credit card transactions are having less credit card fraud on the other hand countries with average credit card transactions are having high rate of credit card fraud. So, in order to avoid the proactive methods are needed. In this a novel machine learning algorithm called cortical algorithm is used. It is the learning algorithm of hierarchical temporal memory and is inspired by the neo cortex of the brain

3.6 Credit Card Fraud Detection: A Fusion Approach using Dempster–Shafer Theory and Bayesian Learning

In these evidences from current as well as past behavior are combined. A fraud detection system is proposed that includes rule based filter, Dempster Shafer adder, transaction history database and Bayesian learner. In rule base the suspicion level of each incoming transaction is determined. Dumpster Shafer is used to combine multiple such evidences and an initial belief is computed. Based on this belief the transactions are classified as normal, abnormal or suspicious. The incoming transactions are initially handled by the rule base using probability values. After this the values are combined using Dumpster Shafer Adder. If the transaction is declared as fraudulent then it is handled by the card holder. If suppose the transaction is suspicious then it is fed in the suspicious table.

3.7 Detecting Credit Card Fraud by Modified Fisher Discriminant Analysis

This employees a linear discriminant called fisher Discriminant. The Linear discriminant is a supervised learning algorithm in which the input region is divided into boundaries called decision boundaries or decision surfaces. The discriminant algorithm is modified to update the weights. This method tries to find the best dimensional hyper plane by which the within class variance is minimized to reduce the overlap and between class variance is maximized. It is a kind of supervised learning algorithm in which the input region is divided into decision regions where the boundaries are called decision boundaries. These boundaries are linear function of input vector x. There is actually a comparison between fisher discriminant and modified fisher discriminant. FDA captures more number of positive cases whereas modified FDA gets more profit by classifying the most profitable transactions. In total FDA can give more number of fraud transactions whereas modified FDA relies on maximizing total profit. This method can label transactions with high usable limits on the cards correctly which leads to prevent losing millions of dollars in real life banking systems. For developing iterative linear discriminant, Linear Perceptron Discriminant function is being used which can solve all the problems related with modified FDA

3.8. Review paper on credit card fraud detection

Suman and Nutan has presented a survey of current techniques used in credit card fraud detection and telecommunication fraud. In this paper, comprehensive review of different techniques to detect fraud is provided. Various types of frauds in this paper include credit card frauds, telecommunication frauds, and computer intrusions, Bankruptcy fraud, Theft fraud/counterfeit fraud, Application fraud, Behavioral fraud. Gass algorithm, Bayesian networks, Hidden markov model, Genetic algorithm, A fusion approach using dempster-shafer theory and Bayesian learning, Decision tree, Neural network and Logistic Regression techniques are explained to detect credit card fraud. One aim of this paper is to identify the user model that best identifies fraud cases.

3.9 Credit card fraud and detection techniques: a review

Delamaire has identified the different types of credit card fraud such as bankruptcy fraud, counterfeit fraud, theft fraud, application fraud and behavioral fraud and review alternative techniques that include pair-wise matching, decision trees, clustering techniques, neural networks, and genetic algorithms. Also state the problems that have been faced by the banks and credit card companies. The next step in this research program is to focus on the implement of a „suspicious" scorecard on a real dataset and its evaluation. The main tasks should be to build scoring models to predict fraudulent behavior, taking into account the fields of behavior that should be related to the different types of credit card fraud identified in this paper, and to evaluate the associated ethical implications. The plan is to take one of the European countries, probably Germany, and then to extend the research to other EU countries.

3.10 Minority report in fraud detection: classification of skewed data

Phua proposed an innovative fraud detection method, built upon existing fraud detection research and Minority Report, to deal with the data mining problem of skewed data distributions. For experiment, Angoss Knowledge Seeker software is used. In this paper, success rates X outperformed all the averaged success rates W by at least 10% on evaluation sets. When applied on the score set, bagged success rates Z performed marginally better than the averaged success rates Y. The future work is to make one classifier more appropriate than another.

3.11 A predictive approach for fraud detection using hidden markov model

Esakkiraj and Chidambaram has design a predictive model with sequence of operations in online transaction by using hidden markov model (HMM) and decides whether the user act as a normal user or fraud user. In the trained system, the new transaction is evaluated with transition and observation probability. Depending upon the observation probability, system finds the acceptance probability and decides whether the transaction should be declined or not. Normally existing fraud detection system for online banking will detect the fraudulent transaction after completion of the transaction. This causes the economic loss and makes the bank name as unsecured. The model predicts the fraudulent during the transaction time and prevents the money transfer. As future work, some effective classification algorithms instead of using clustering which can perform well for the prediction.

4. EXISITING SYSTEM:

Previously they used the most common methods like rule induction techniques, fuzzy system, decision trees, Support Vector Machines (SVM), K-Nearest Neighbor algorithms to detect the fraud transaction using a credit card. From our perspective, neural networks will generate more accurate results.

5. PROPOSED SYSTEM:

To increase the accuracy and precision we use the algorithms Logistic Regression, Convolution

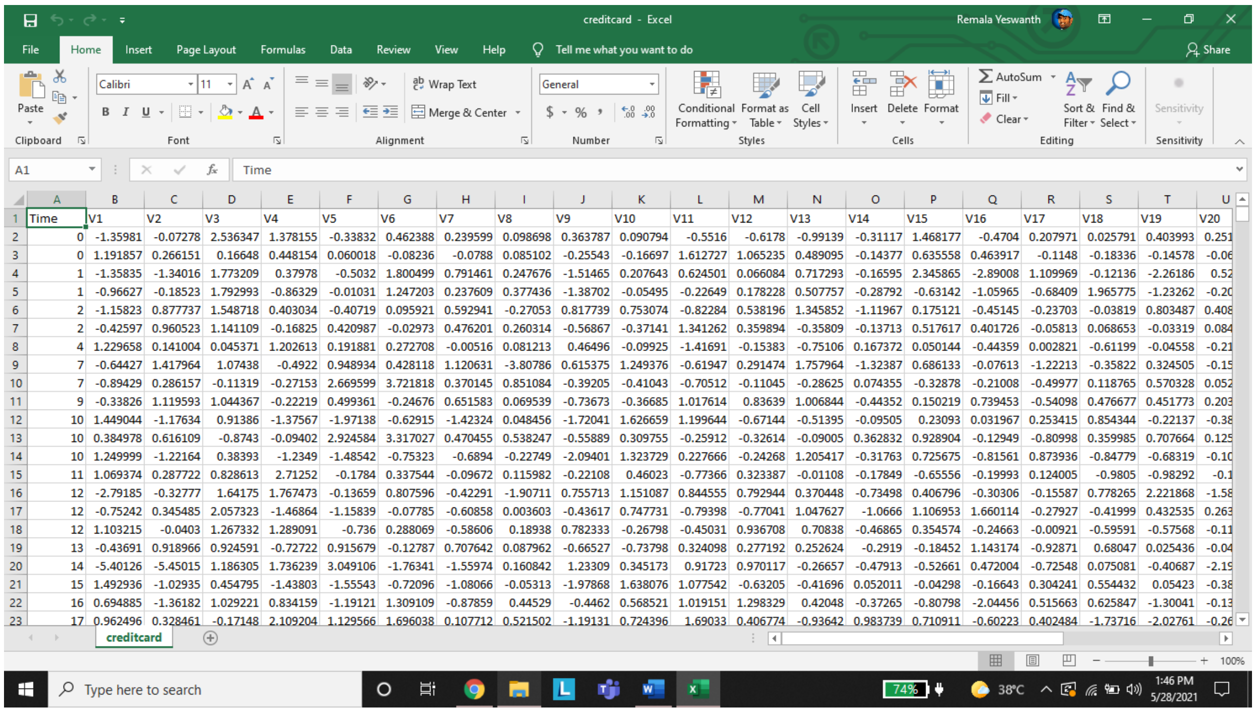
Neural Networks. Logistic Regression is a statistical model that tries to minimize the cost of how wrong a prediction is. CNN algorithm is used, to capture the intrinsic patterns of fraud behaviors learned from labeled data. So will make use of accuracy and precision to evaluate the performance of the proposed system.

Convolution Neural Networks algorithms are used to achieve high accuracy rates and to increase the detection process. Convolution Neural networks are used to identify the underlying patterns that are followed in the previous cases and thereby increasing efficiency.

6. DATASET:

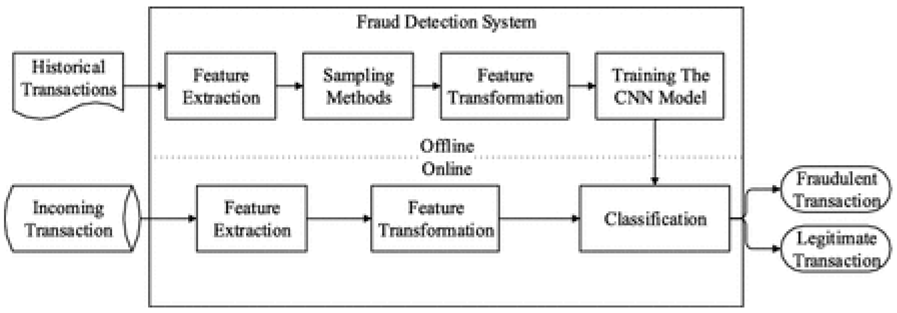
https://www.kaggle.com/mlg-ulb/creditcardfraud

- The datasets contain transactions made by credit cards in September 2013 by European cardholders.
- This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
- It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset.
- The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning.
- Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.



Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791	0.403993	0.251
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336	-0.14578	-0.06
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643	0.624501	0.066084	0.717293	-0.16595	2.345865	-2.89008	1.109969	-0.12136	-2.26186	0.52
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.178228	0.507757	-0.28792	-0.63142	-1.05965	-0.68409	1.965775	-1.23262	-0.20
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.538196	1.345852	-1.11967	0.175121	-0.45145	-0.23703	-0.03819	0.803487	0.408
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.359894	-0.35809	-0.13713	0.517617	0.401726	-0.05813	0.068653	-0.03319	0.084
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.167372	0.050144	-0.44359	0.002821	-0.61199	-0.04558	-0.21
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376	-0.61947	0.291474	1.757964	-1.32387	0.686133	-0.07613	-1.22213	-0.35822	0.324505	-0.15
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.074355	-0.32878	-0.21008	-0.49977	0.118765	0.570328	0.052
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.83639	1.006844	-0.44352	0.150219	0.739453	-0.54098	0.476677	0.451773	0.203
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659	1.199644	-0.67144	-0.51395	-0.09505	0.23093	0.031967	0.253415	0.854344	-0.22137	-0.38
10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755	-0.25912	-0.32614	-0.09005	0.362832	0.928904	-0.12949	-0.80998	0.359985	0.707664	0.125
10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.323729	0.227666	-0.24268	1.205417	-0.31763	0.725675	-0.81561	0.873936	-0.84779	-0.68319	-0.10
11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023	-0.77366	0.323387	-0.01108	-0.17849	-0.65556	-0.19993	0.124005	-0.9805	-0.98292	-0.1
12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087	0.844555	0.792944	0.370448	-0.73498	0.406796	-0.30306	-0.15587	0.778265	2.221868	-1.58
12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617	0.747731	-0.79398	-0.77041	1.047627	-1.0666	1.106953	1.660114	-0.27927	-0.41999	0.432535	0.263
12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333	-0.26798	-0.45031	0.936708	0.70838	-0.46865	0.354574	-0.24663	-0.00921	-0.59591	-0.57568	-0.11
13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527	-0.73798	0.324098	0.277192	0.252624	-0.2919	-0.18452	1.143174	-0.92871	0.68047	0.025436	-0.04
14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.55974	0.160842	1.23309	0.345173	0.91723	0.970117	-0.26657	-0.47913	-0.52661	0.472004	-0.72548	0.075081	-0.40687	-2.19
15	1.492936	-1.02935	0.454795	-1.43803	-1.55543	-0.72096	-1.08066	-0.05313	-1.97868	1.638076	1.077542	-0.63205	-0.41696	0.052011	-0.04298	-0.16643	0.304241	0.554432	0.05423	-0.38
16	0.694885	-1.36182	1.029221	0.834159	-1.19121	1.309109	-0.87859	0.44529	-0.4462	0.568521	1.019151	1.298329	0.42048	-0.37265	-0.80798	-2.04456	0.515663	0.625847	-1.30041	-0.13
17	0.962496	0.328461	-0.17148	2.109204	1.129566	1.696038	0.107712	0.521502	-1.19131	0.724396	1.69033	0.406774	-0.93642	0.983739	0.710911	-0.60223	0.402484	-1.73716	-2.02761	-0.26

7. ARCHITECTURE DIAGRAM:



8. CODE:

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
```

```
from scipy import stats
import tensorflow as tf
from tensorflow import keras
import seaborn as sns


# In[ ]:


from tensorflow.python.keras.layers import Conv2D, MaxPooling2D, ReLU


# In[ ]:


def cnn_model(img_rows, img_cols, img_channels):
    model = Sequential()
    model.add(Conv2D(64, (3, 3),activation='linear',kernel_initializer='he_uniform',
                     input_shape=(img_rows, img_cols, img_channels)))
    model.add(ReLU())   # add an advanced activation
    model.add(MaxPooling2D(pool_size=(5, 5)))
    model.add(Conv2D(32, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(ReLU())   # add an advanced activation
    model.add(MaxPooling2D(pool_size=(3, 3)))
    model.add(Conv2D(16, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(ReLU())   # add an advanced activation
    model.add(MaxPooling2D(pool_size=(3, 3)))
    model.add(Flatten())
    model.add(Dense(1024))
    model.add(Dense(1024))
    model.add(ReLU())   # add an advanced activation
    model.add(Dense(4))
    model.add(Activation('softmax'))

    return model


# In[ ]:


from pylab import rcParams
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.manifold import TSNE
from sklearn.metrics import classification_report, accuracy_score


# In[ ]:


from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.applications.vgg16 import decode_predictions
from tensorflow.keras.applications.vgg16 import VGG16


# In[ ]:


from tensorflow.python.keras.layers import Input, Dense
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard
from tensorflow.keras import regularizers, Sequential


# In[ ]:


get_ipython().run_line_magic('matplotlib', 'inline')
sns.set(style='whitegrid', palette='muted', font_scale=1.5)
rcParams['figure.figsize'] = 14, 8
RANDOM_SEED = 42


# In[ ]:


model = tf.keras.Sequential([
    tf.keras.layers.Dense(5, input_shape=(3,)),
    tf.keras.layers.Softmax()])
model.save('/tmp/model')
loaded_model = tf.keras.models.load_model('/tmp/model')
x = tf.random.uniform((10, 3))
assert np.allclose(model.predict(x), loaded_model.predict(x))


# In[ ]:


LABELS = ["Normal", "Fraud"]
```

```
# In[ ]:


df = pd.read_csv("creditcard.csv")
df.head()


# In[ ]:


df.shape


# In[ ]:


df.isnull().values.any()


# In[ ]:


count_classes = pd.value_counts(df['Class'], sort = True)
count_classes.plot(kind = 'bar', rot=0)
plt.title("Transaction class distribution")
plt.xticks(range(2), LABELS)
plt.xlabel("Class")
plt.ylabel("Frequency")


# In[ ]:
```

```
frauds = df[df.Class == 1]
normal = df[df.Class == 0]
frauds.shape


# In[ ]:


normal.shape


# In[ ]:


frauds.Amount.describe()


# In[ ]:


normal.Amount.describe()


# In[ ]:


f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Amount per transaction by class')
```

```
bins = 50

ax1.hist(frauds.Amount, bins = bins)
ax1.set_title('Fraud')

ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')

plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show()



# In[ ]:



f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Time of transaction vs Amount by class')

ax1.scatter(frauds.Time, frauds.Amount)
ax1.set_title('Fraud')

ax2.scatter(normal.Time, normal.Amount)
ax2.set_title('Normal')

plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```

```
# In[ ]:


data = df.drop(['Time'], axis=1)


# In[ ]:


from sklearn.preprocessing import StandardScaler

data['Amount'] = StandardScaler().fit_transform(data['Amount'].values.reshape(-1, 1))


# In[ ]:


non_fraud = data[data['Class'] == 0] #.sample(1000)
fraud = data[data['Class'] == 1]

df = non_fraud.append(fraud).sample(frac=1).reset_index(drop=True)
X = df.drop(['Class'], axis = 1).values
Y = df["Class"].values


# In[ ]:


X_train, X_test = train_test_split(data, test_size=0.2, random_state=RANDOM_SEED)
X_train_fraud = X_train[X_train.Class == 1]
```

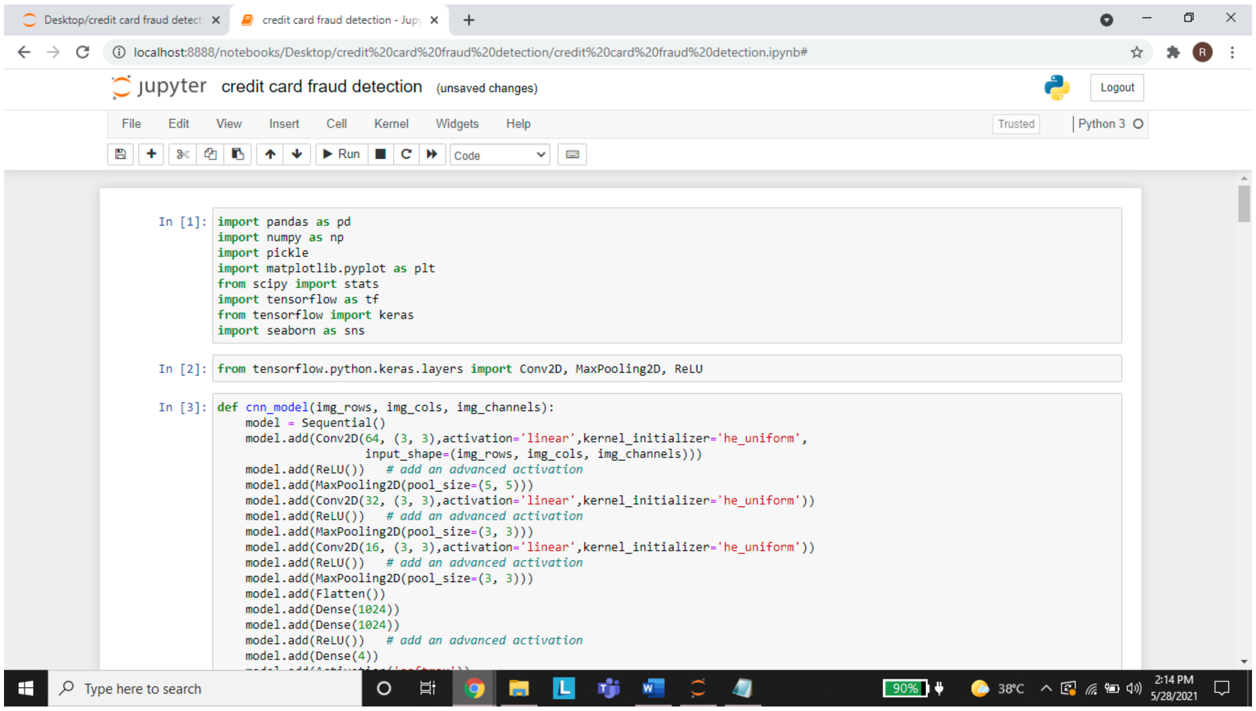
```
X_train = X_train[X_train.Class == 0]
X_train = X_train.drop(['Class'], axis=1)
y_test = X_test['Class']
X_test = X_test.drop(['Class'], axis=1)
X_train = X_train.values
X_test = X_test.values
X_train.shape


# In[ ]:



input_layer = Input(shape=(X.shape[1],))

## encoding part
encoded = Dense(100, activation='tanh', activity_regularizer=regularizers.l1(10e-5))(input_layer)
encoded = Dense(50, activation='relu')(encoded)

## decoding part
decoded = Dense(50, activation='tanh')(encoded)
decoded = Dense(100, activation='tanh')(decoded)

## output layer
output_layer = Dense(X.shape[1], activation='relu')(decoded)


# In[ ]:



model.save('model.h5')
```

```
# In[ ]:


from tensorflow.keras import models


# In[ ]:


from tensorflow.keras import models


# In[ ]:


from tensorflow.keras.models import Model, load_model


# In[ ]:


autoencoder = Model(input_layer, output_layer)
autoencoder.compile(optimizer="adadelta", loss="mse")


# In[ ]:


x = data.drop(["Class"], axis=1)
y = data["Class"].values
```

```
x_scale = MinMaxScaler().fit_transform(x.values)
x_norm, x_fraud = x_scale[y == 0], x_scale[y == 1]

autoencoder.fit(x_norm[0:2000], x_norm[0:2000],
                batch_size = 256, epochs = 10,
                shuffle = True, validation_split = 0.20);



# In[ ]:



hidden_representation = Sequential()
hidden_representation.add(autoencoder.layers[0])
hidden_representation.add(autoencoder.layers[1])
hidden_representation.add(autoencoder.layers[2])



# In[ ]:



norm_hid_rep = hidden_representation.predict(x_norm[:3000])
fraud_hid_rep = hidden_representation.predict(x_fraud)



# In[ ]:



rep_x = np.append(norm_hid_rep, fraud_hid_rep, axis = 0)
y_n = np.zeros(norm_hid_rep.shape[0])
y_f = np.ones(fraud_hid_rep.shape[0])
```

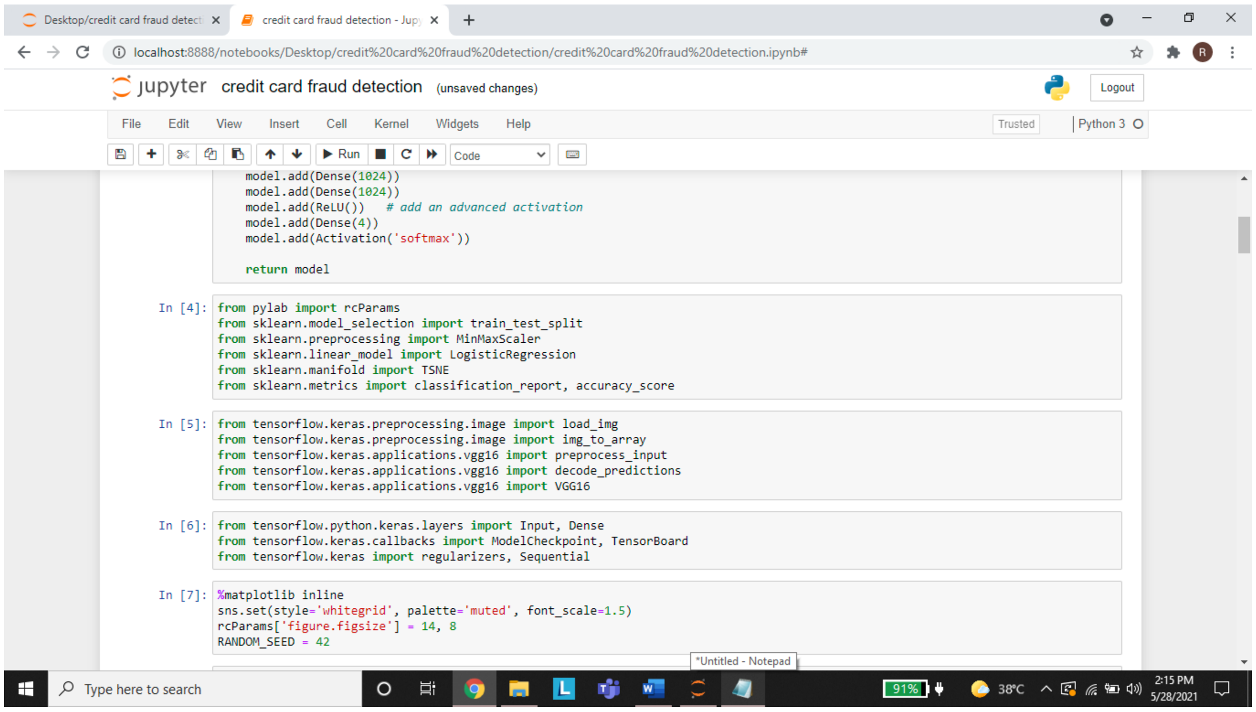
```
rep_y = np.append(y_n, y_f)


# In[ ]:


train_x, val_x, train_y, val_y = train_test_split(rep_x, rep_y, test_size=0.25)


# In[ ]:


clf = LogisticRegression(solver="lbfgs").fit(train_x, train_y)
pred_y = clf.predict(val_x)

print ("")
print ("Classification Report: ")
print (classification_report(val_y, pred_y))

print ("")
print ("Accuracy Score: ", accuracy_score(val_y, pred_y))
```

9. RESULT:



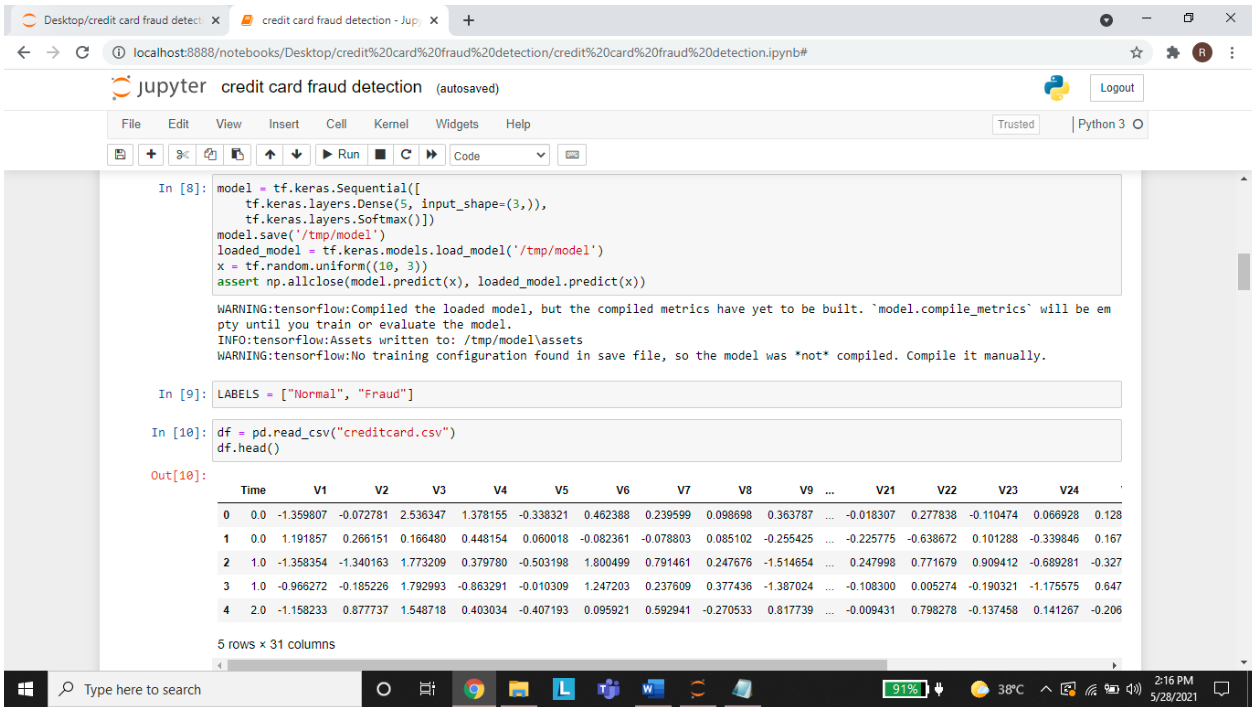
Desktop/credit card fraud detect
credit card fraud detection - Jup
localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Type here to search
90%
38°C
2:14 PM
5/28/2021

```
In [1]: import pandas as pd
        import numpy as np
        import pickle
        import matplotlib.pyplot as plt
        from scipy import stats
        import tensorflow as tf
        from tensorflow import keras
        import seaborn as sns

In [2]: from tensorflow.python.keras.layers import Conv2D, MaxPooling2D, ReLU

In [3]: def cnn_model(img_rows, img_cols, img_channels):
            model = Sequential()
            model.add(Conv2D(64, (3, 3),activation='linear',kernel_initializer='he_uniform',
                             input_shape=(img_rows, img_cols, img_channels)))
            model.add(ReLU())   # add an advanced activation
            model.add(MaxPooling2D(pool_size=(5, 5)))
            model.add(Conv2D(32, (3, 3),activation='linear',kernel_initializer='he_uniform'))
            model.add(ReLU())   # add an advanced activation
            model.add(MaxPooling2D(pool_size=(3, 3)))
            model.add(Conv2D(16, (3, 3),activation='linear',kernel_initializer='he_uniform'))
            model.add(ReLU())   # add an advanced activation
            model.add(MaxPooling2D(pool_size=(3, 3)))
            model.add(Flatten())
            model.add(Dense(1024))
            model.add(Dense(1024))
            model.add(ReLU())   # add an advanced activation
            model.add(Dense(4))
```

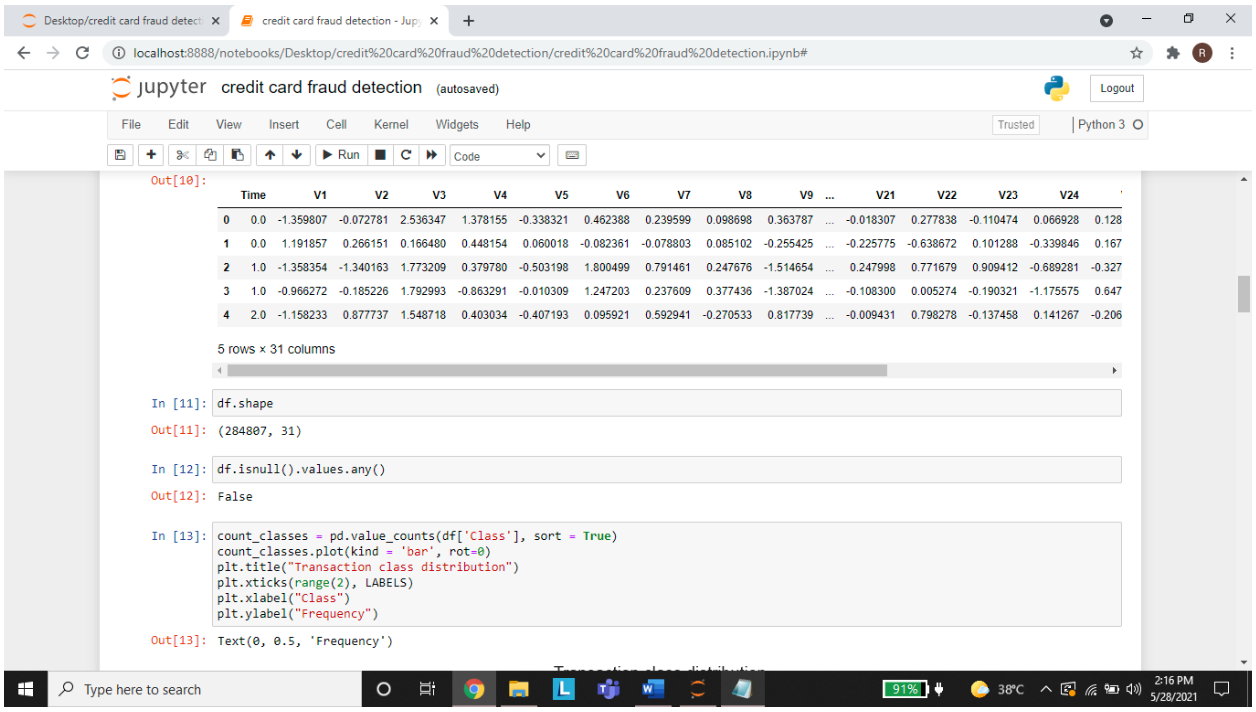


Desktop/credit card fraud detect
credit card fraud detection - Jup
localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
*Untitled - Notepad
Type here to search
91%
38°C
2:15 PM
5/28/2021

```
            model.add(Dense(1024))
            model.add(Dense(1024))
            model.add(ReLU())   # add an advanced activation
            model.add(Dense(4))
            model.add(Activation('softmax'))

            return model

In [4]: from pylab import rcParams
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.linear_model import LogisticRegression
        from sklearn.manifold import TSNE
        from sklearn.metrics import classification_report, accuracy_score

In [5]: from tensorflow.keras.preprocessing.image import load_img
        from tensorflow.keras.preprocessing.image import img_to_array
        from tensorflow.keras.applications.vgg16 import preprocess_input
        from tensorflow.keras.applications.vgg16 import decode_predictions
        from tensorflow.keras.applications.vgg16 import VGG16

In [6]: from tensorflow.python.keras.layers import Input, Dense
        from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard
        from tensorflow.keras import regularizers, Sequential

In [7]: %matplotlib inline
        sns.set(style='whitegrid', palette='muted', font_scale=1.5)
        rcParams['figure.figsize'] = 14, 8
        RANDOM_SEED = 42
```

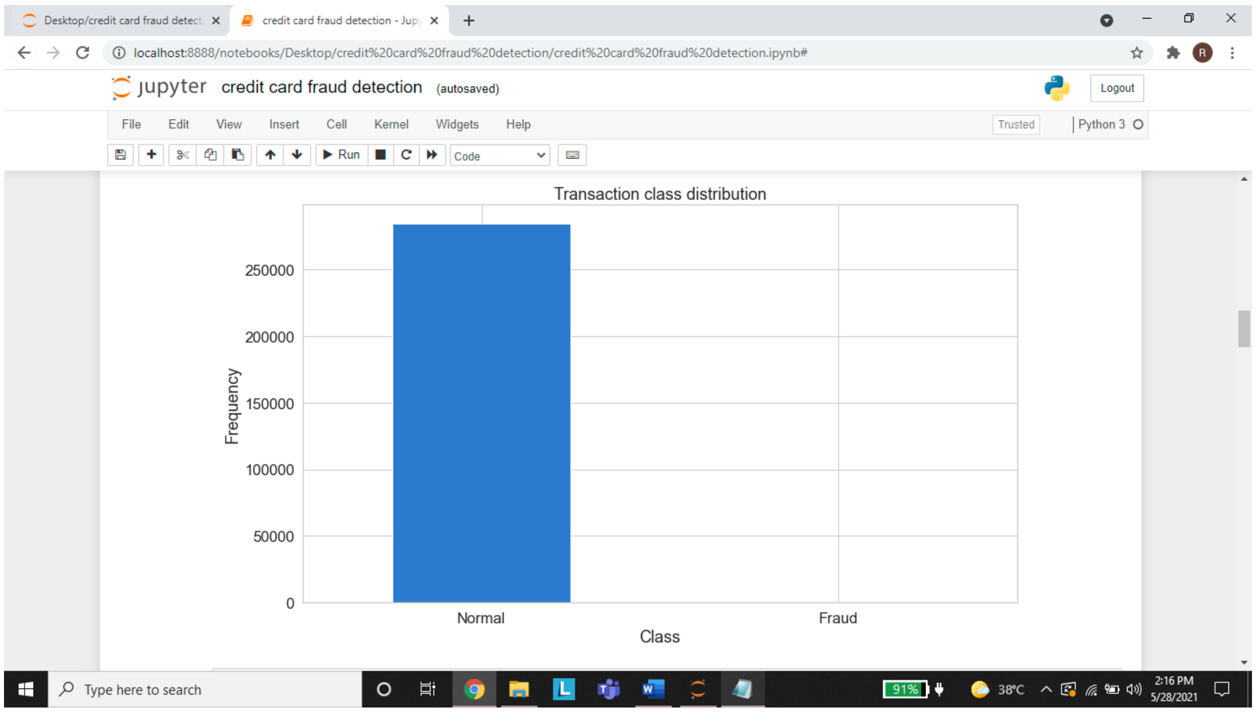

Desktop/credit card fraud detect
credit card fraud detection - Jup
localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Type here to search
91%
38°C
2:16 PM
5/28/2021

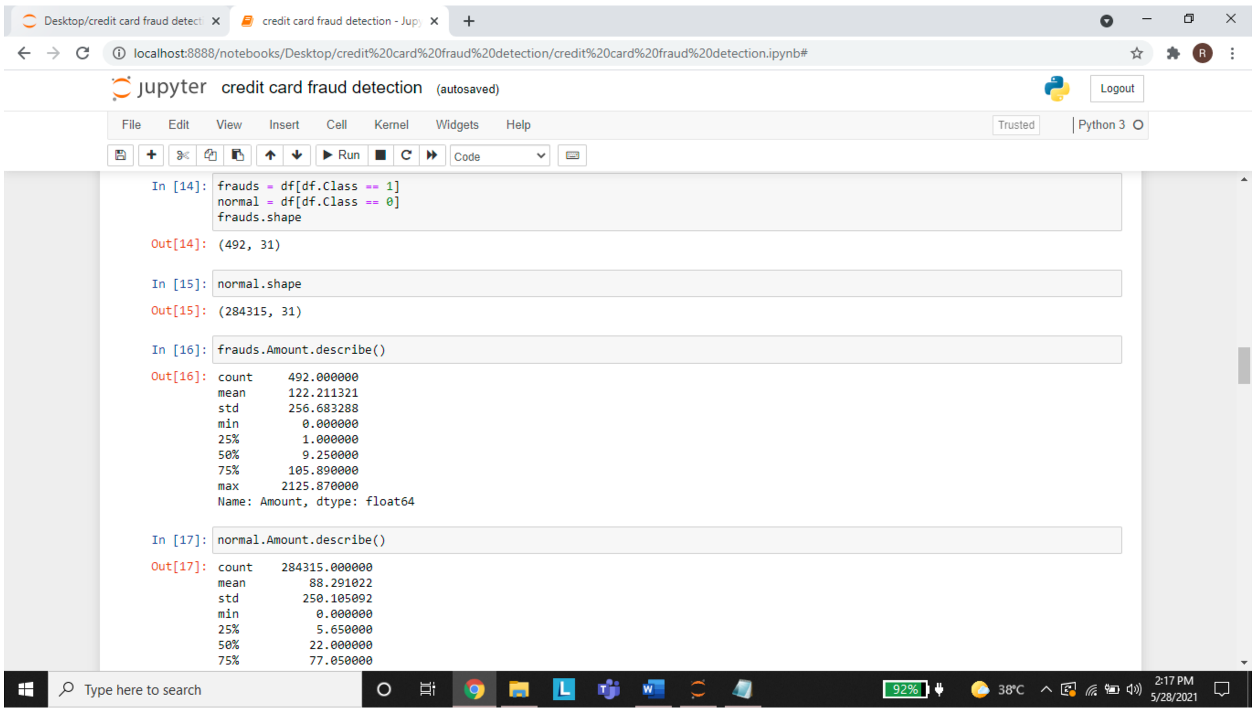
```
In [8]: model = tf.keras.Sequential([
            tf.keras.layers.Dense(5, input_shape=(3,)),
            tf.keras.layers.Softmax()])
        model.save('/tmp/model')
        loaded_model = tf.keras.models.load_model('/tmp/model')
        x = tf.random.uniform((10, 3))
        assert np.allclose(model.predict(x), loaded_model.predict(x))

        WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be em
        pty until you train or evaluate the model.
        INFO:tensorflow:Assets written to: /tmp/model\assets
        WARNING:tensorflow:No training configuration found in save file, so the model was *not* compiled. Compile it manually.

In [9]: LABELS = ["Normal", "Fraud"]

In [10]: df = pd.read_csv("creditcard.csv")
         df.head()

Out[10]:
     Time        V1        V2        V3        V4        V5        V6        V7        V8        V9  ...       V21       V22       V23       V24
  0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128
  1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167
  2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327
  3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647
  4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206

5 rows × 31 columns
```


Desktop/credit card fraud detect
credit card fraud detection - Jup
localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Type here to search
91%
38°C
2:16 PM
5/28/2021

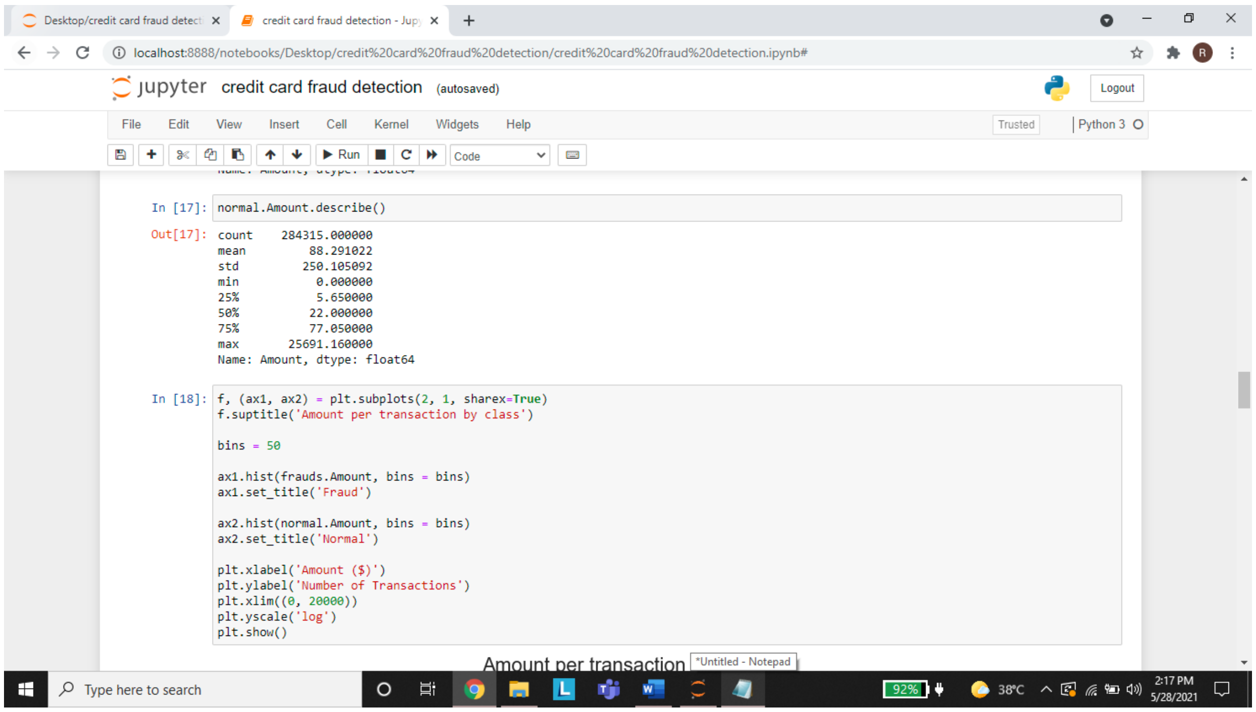
```
Out[10]:
     Time        V1        V2        V3        V4        V5        V6        V7        V8        V9  ...       V21       V22       V23       V24
  0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128
  1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167
  2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327
  3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647
  4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206

5 rows × 31 columns
```

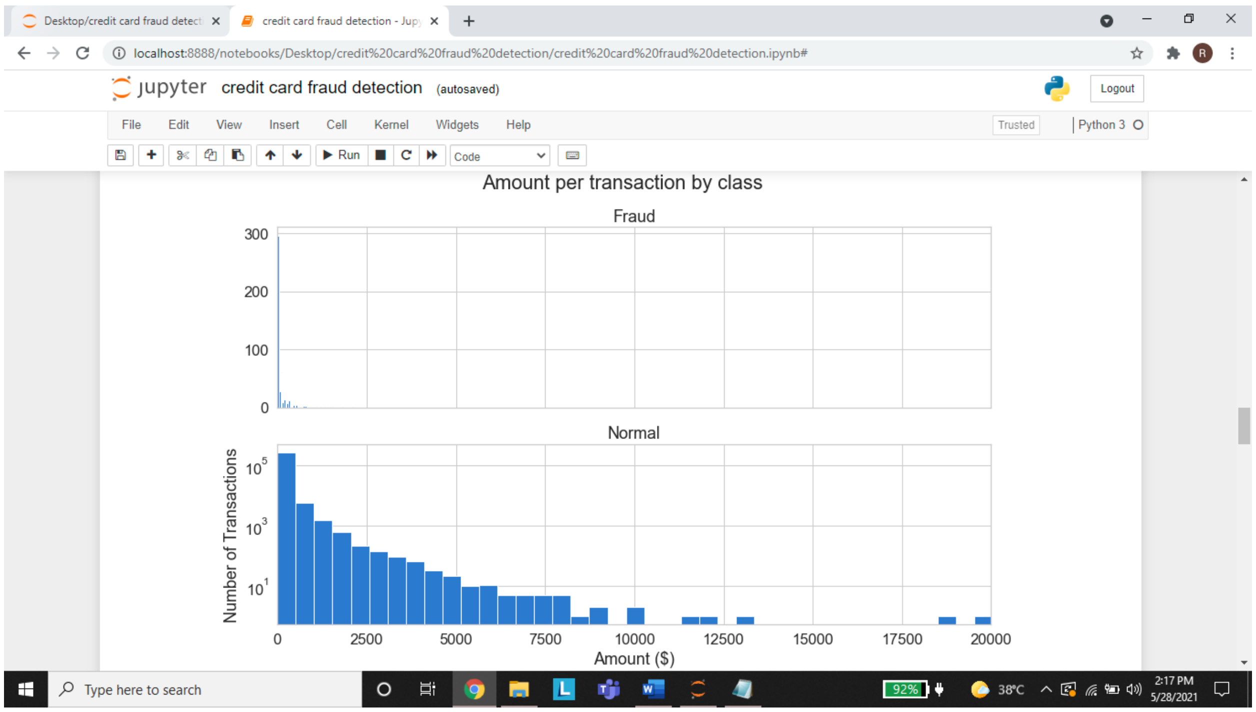
```
In [11]: df.shape

Out[11]: (284807, 31)

In [12]: df.isnull().values.any()

Out[12]: False

In [13]: count_classes = pd.value_counts(df['Class'], sort = True)
         count_classes.plot(kind = 'bar', rot=0)
         plt.title("Transaction class distribution")
         plt.xticks(range(2), LABELS)
         plt.xlabel("Class")
         plt.ylabel("Frequency")

Out[13]: Text(0, 0.5, 'Frequency')
```

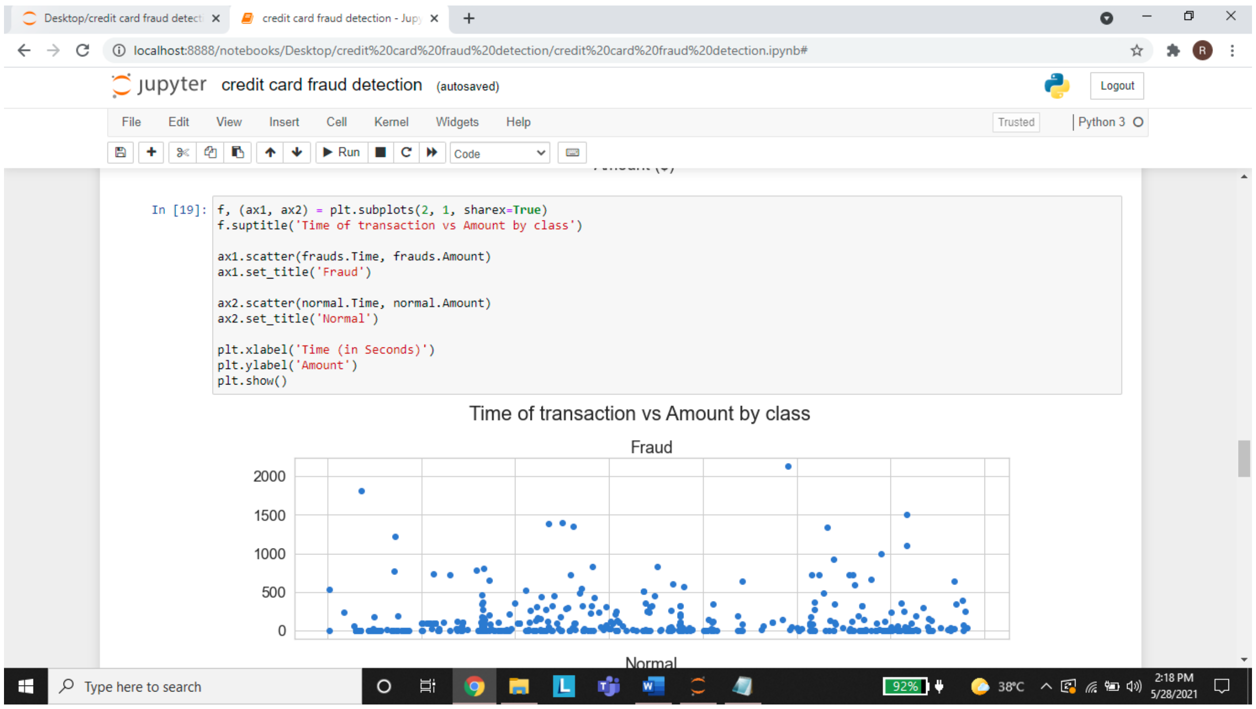




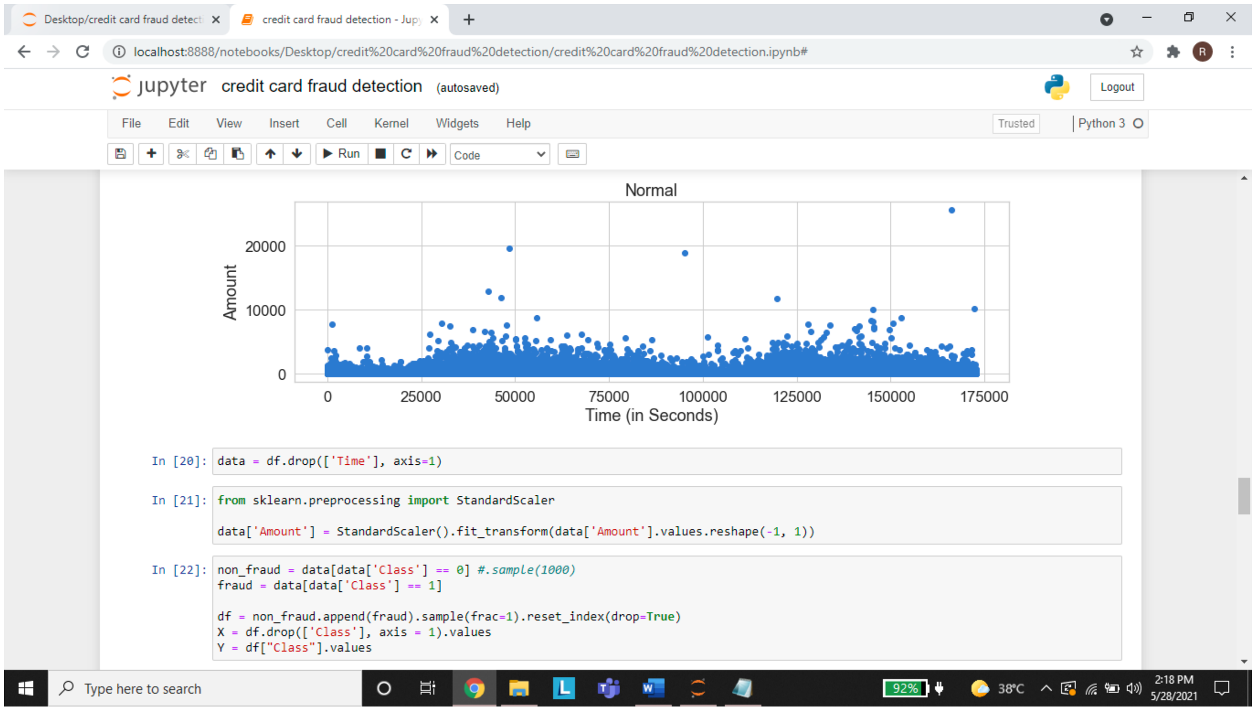
```
In [14]: frauds = df[df.Class == 1]
         normal = df[df.Class == 0]
         frauds.shape

Out[14]: (492, 31)

In [15]: normal.shape

Out[15]: (284315, 31)

In [16]: frauds.Amount.describe()

Out[16]: 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

In [17]: normal.Amount.describe()

Out[17]: count    284315.000000
         mean         88.291022
         std         250.105092
         min           0.000000
         25%           5.650000
         50%          22.000000
         75%          77.050000
```


```
In [17]: normal.Amount.describe()

Out[17]: 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

In [18]: f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
         f.suptitle('Amount per transaction by class')

         bins = 50

         ax1.hist(frauds.Amount, bins = bins)
         ax1.set_title('Fraud')

         ax2.hist(normal.Amount, bins = bins)
         ax2.set_title('Normal')

         plt.xlabel('Amount ($)')
         plt.ylabel('Number of Transactions')
         plt.xlim((0, 20000))
         plt.yscale('log')
         plt.show()
```

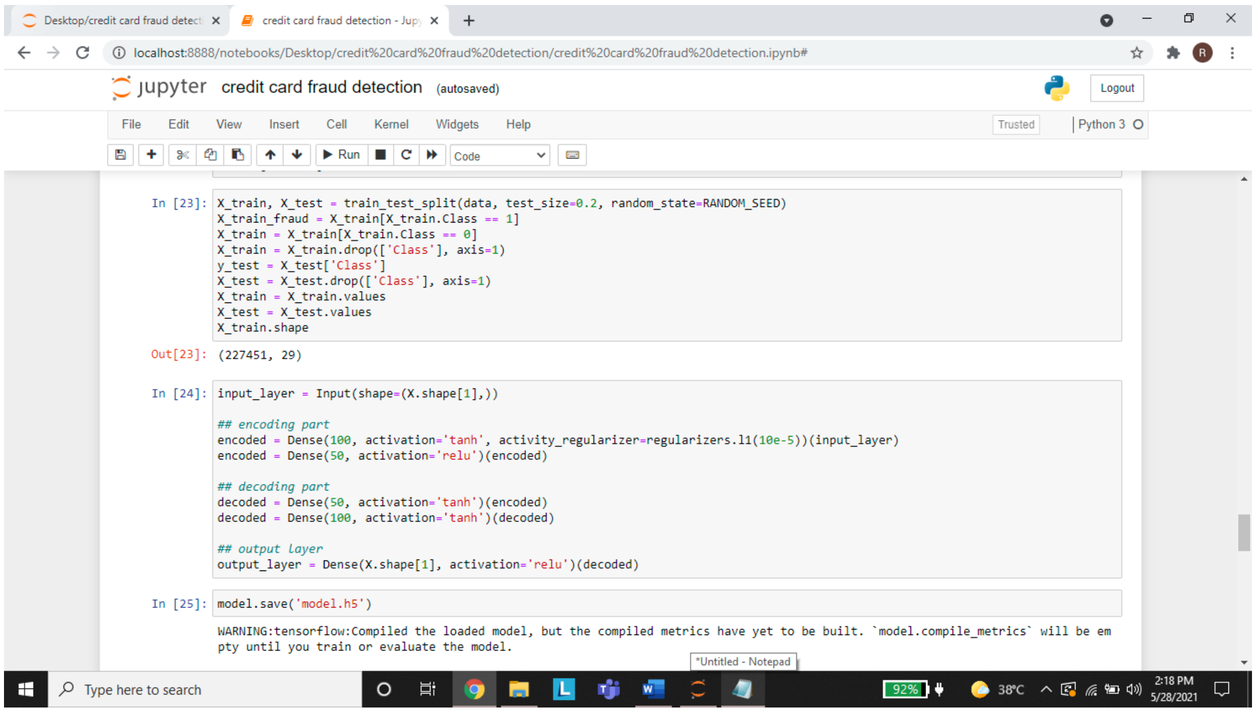




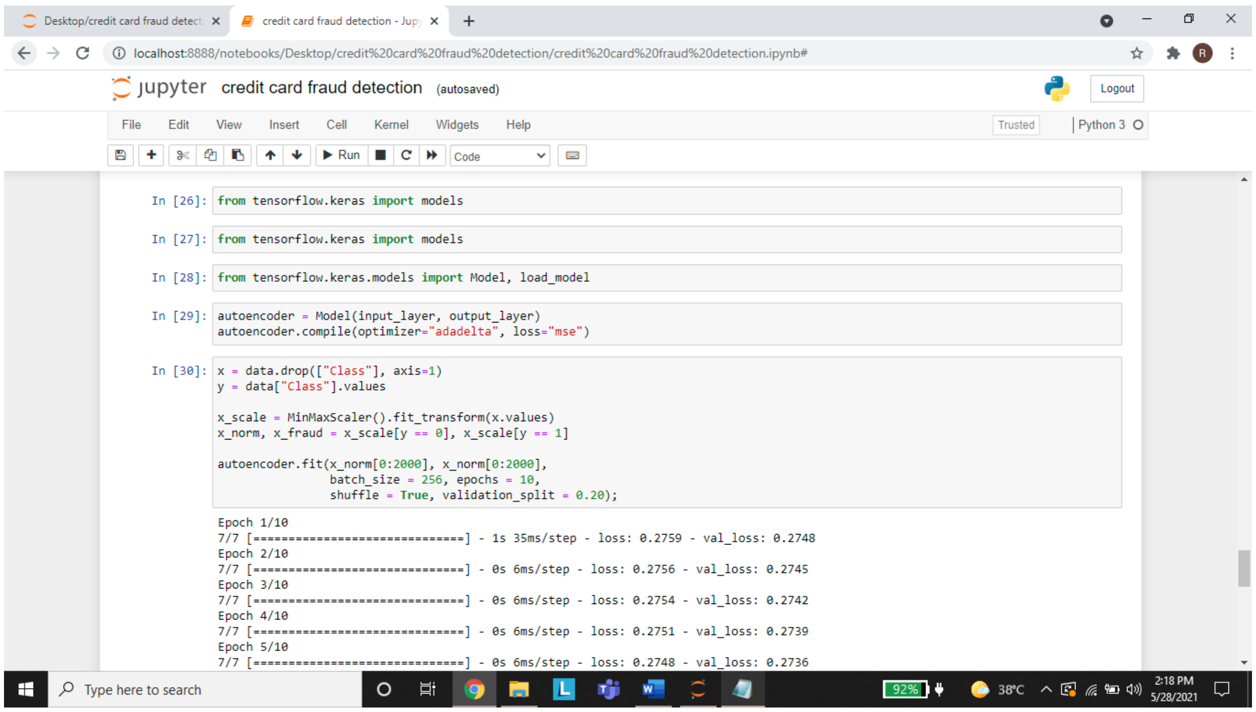
```
In [19]: f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
         f.suptitle('Time of transaction vs Amount by class')

         ax1.scatter(frauds.Time, frauds.Amount)
         ax1.set_title('Fraud')

         ax2.scatter(normal.Time, normal.Amount)
         ax2.set_title('Normal')

         plt.xlabel('Time (in Seconds)')
         plt.ylabel('Amount')
         plt.show()
```



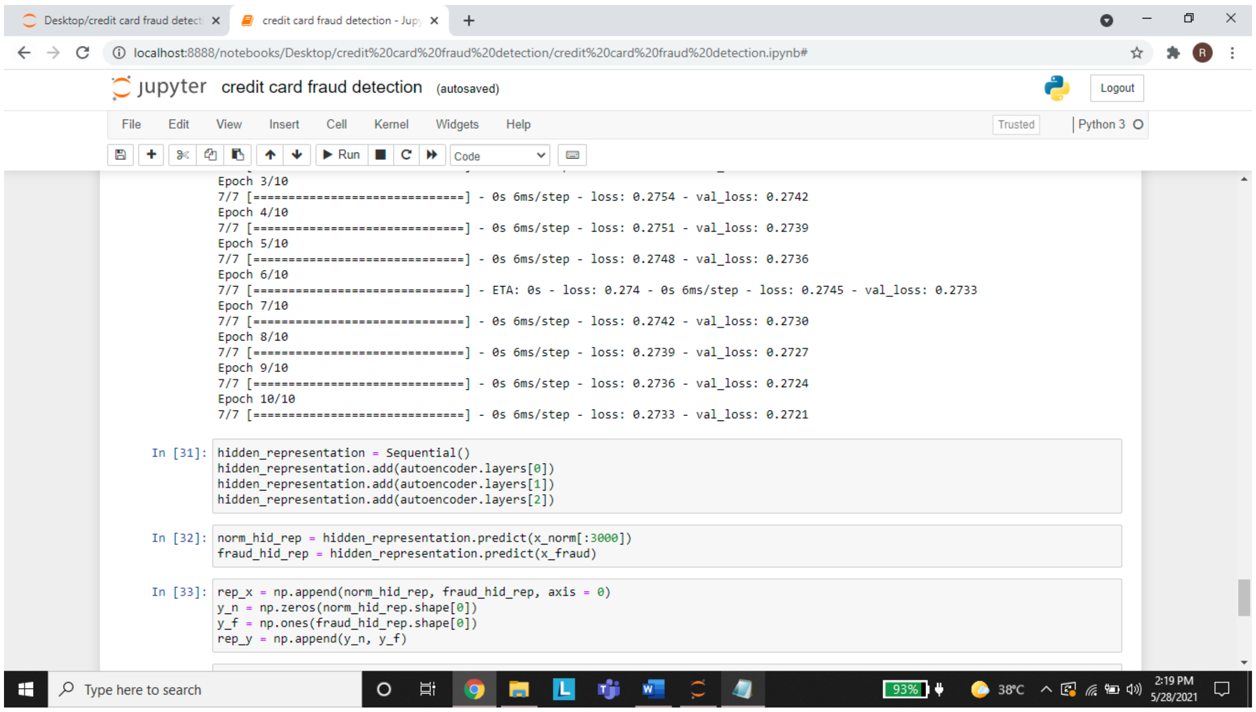
```
In [20]: data = df.drop(['Time'], axis=1)

In [21]: from sklearn.preprocessing import StandardScaler

         data['Amount'] = StandardScaler().fit_transform(data['Amount'].values.reshape(-1, 1))

In [22]: non_fraud = data[data['Class'] == 0] #.sample(1000)
         fraud = data[data['Class'] == 1]

         df = non_fraud.append(fraud).sample(frac=1).reset_index(drop=True)
         X = df.drop(['Class'], axis = 1).values
         Y = df["Class"].values
```


Desktop/credit card fraud detect
credit card fraud detection - Jup
localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
*Untitled - Notepad
Type here to search
92%
38°C
2:18 PM
5/28/2021

```
In [23]: X_train, X_test = train_test_split(data, test_size=0.2, random_state=RANDOM_SEED)
         X_train_fraud = X_train[X_train.Class == 1]
         X_train = X_train[X_train.Class == 0]
         X_train = X_train.drop(['Class'], axis=1)
         y_test = X_test['Class']
         X_test = X_test.drop(['Class'], axis=1)
         X_train = X_train.values
         X_test = X_test.values
         X_train.shape

Out[23]: (227451, 29)

In [24]: input_layer = Input(shape=(X.shape[1],))

         ## encoding part
         encoded = Dense(100, activation='tanh', activity_regularizer=regularizers.l1(10e-5))(input_layer)
         encoded = Dense(50, activation='relu')(encoded)

         ## decoding part
         decoded = Dense(50, activation='tanh')(encoded)
         decoded = Dense(100, activation='tanh')(decoded)

         ## output layer
         output_layer = Dense(X.shape[1], activation='relu')(decoded)

In [25]: model.save('model.h5')

         WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be em
         pty until you train or evaluate the model.
```


Desktop/credit card fraud detect
credit card fraud detection - Jup
localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Type here to search
92%
38°C
2:18 PM
5/28/2021

```
In [26]: from tensorflow.keras import models

In [27]: from tensorflow.keras import models

In [28]: from tensorflow.keras.models import Model, load_model

In [29]: autoencoder = Model(input_layer, output_layer)
         autoencoder.compile(optimizer="adadelta", loss="mse")

In [30]: x = data.drop(["Class"], axis=1)
         y = data["Class"].values

         x_scale = MinMaxScaler().fit_transform(x.values)
         x_norm, x_fraud = x_scale[y == 0], x_scale[y == 1]

         autoencoder.fit(x_norm[0:2000], x_norm[0:2000],
                         batch_size = 256, epochs = 10,
                         shuffle = True, validation_split = 0.20);

         Epoch 1/10
         7/7 [==============================] - 1s 35ms/step - loss: 0.2759 - val_loss: 0.2748
         Epoch 2/10
         7/7 [==============================] - 0s 6ms/step - loss: 0.2756 - val_loss: 0.2745
         Epoch 3/10
         7/7 [==============================] - 0s 6ms/step - loss: 0.2754 - val_loss: 0.2742
         Epoch 4/10
         7/7 [==============================] - 0s 6ms/step - loss: 0.2751 - val_loss: 0.2739
         Epoch 5/10
         7/7 [==============================] - 0s 6ms/step - loss: 0.2748 - val_loss: 0.2736
```

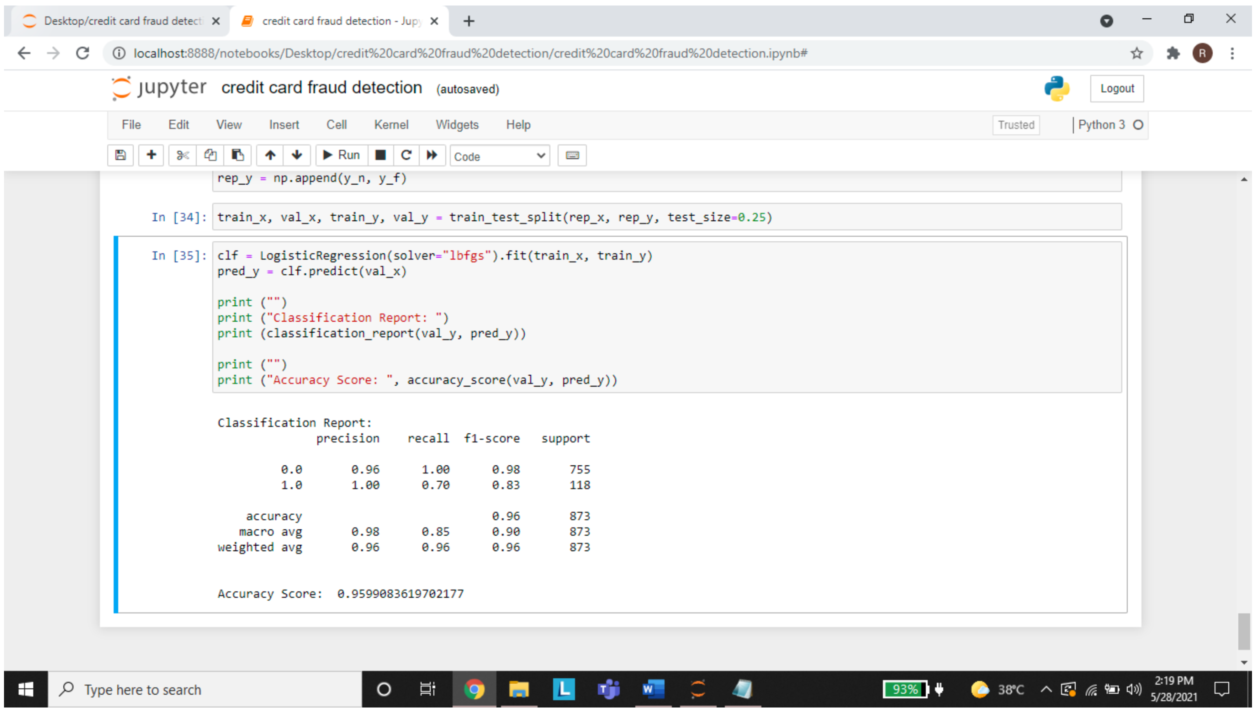

localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3

```
Epoch 3/10
7/7 [==============================] - 0s 6ms/step - loss: 0.2754 - val_loss: 0.2742
Epoch 4/10
7/7 [==============================] - 0s 6ms/step - loss: 0.2751 - val_loss: 0.2739
Epoch 5/10
7/7 [==============================] - 0s 6ms/step - loss: 0.2748 - val_loss: 0.2736
Epoch 6/10
7/7 [==============================] - ETA: 0s - loss: 0.274 - 0s 6ms/step - loss: 0.2745 - val_loss: 0.2733
Epoch 7/10
7/7 [==============================] - 0s 6ms/step - loss: 0.2742 - val_loss: 0.2730
Epoch 8/10
7/7 [==============================] - 0s 6ms/step - loss: 0.2739 - val_loss: 0.2727
Epoch 9/10
7/7 [==============================] - 0s 6ms/step - loss: 0.2736 - val_loss: 0.2724
Epoch 10/10
7/7 [==============================] - 0s 6ms/step - loss: 0.2733 - val_loss: 0.2721
```

```
In [31]: hidden_representation = Sequential()
         hidden_representation.add(autoencoder.layers[0])
         hidden_representation.add(autoencoder.layers[1])
         hidden_representation.add(autoencoder.layers[2])
```

```
In [32]: norm_hid_rep = hidden_representation.predict(x_norm[:3000])
         fraud_hid_rep = hidden_representation.predict(x_fraud)
```

```
In [33]: rep_x = np.append(norm_hid_rep, fraud_hid_rep, axis = 0)
         y_n = np.zeros(norm_hid_rep.shape[0])
         y_f = np.ones(fraud_hid_rep.shape[0])
         rep_y = np.append(y_n, y_f)
```


localhost:8888/notebooks/Desktop/credit%20card%20fraud%20detection/credit%20card%20fraud%20detection.ipynb#
jupyter credit card fraud detection (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3

```
         rep_y = np.append(y_n, y_f)
```

```
In [34]: train_x, val_x, train_y, val_y = train_test_split(rep_x, rep_y, test_size=0.25)
```

```
In [35]: clf = LogisticRegression(solver="lbfgs").fit(train_x, train_y)
         pred_y = clf.predict(val_x)

         print ("")
         print ("Classification Report: ")
         print (classification_report(val_y, pred_y))

         print ("")
         print ("Accuracy Score: ", accuracy_score(val_y, pred_y))
```

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.96      1.00      0.98       755
         1.0       1.00      0.70      0.83       118

    accuracy                           0.96       873
   macro avg       0.98      0.85      0.90       873
weighted avg       0.96      0.96      0.96       873


Accuracy Score:  0.9599083619702177
```

10. CONCLUSION:

The CNN model based on feature rearrangement constructed in this paper has an excellent experimental, performance with a good stability. The model needs neither high dimensional input features nor derivative variables and can find a relatively good ordered arrangement of input within a certain number of times. Compared with most existing CNN model, this model saves much calculation time of the derived variables, which makes the design and adjustment process of the model quick and easy. And there is a higher level of availability in an environment where online transactions require rapid response and accurate identification.

11. REFERENCES:

[1] Aswathy M S, Liji Sameul "Survey on Credit Card Fraud Detection".

[2] Y. Sahin, S. Bulkan, and E. Duman, ``A cost-sensitive decision tree approach for fraud detection," Expert Syst. Appl., vol. 40, no. 15, pp. 5916_5923, 2013.

[3] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar, ``Credit card fraud detection using hidden Markov model," IEEE Trans. Depend. Sec. Comput., vol. 5, no. 1, pp. 37, Jan. 2008.

[4] J. T. Quah and M. Sriganesh, ``Real-time credit card fraud detection using computational intelligence," Expert Syst. Appl., vol. 35, no. 4, pp..

[5] S. Panigrahi, A. Kundu, S. Sural, and A. K. Majumdar, ``Credit card fraud detection: A fusion approach using Dempster Shafer theory and Bayesian learning," Inf. Fusion, vol. 10, no. 4.

[6] N. Mahmoudi and E. Duman, ``Detecting credit card fraud by modified fisher discriminant analysis," Expert Syst. Appl., vol. 42, no. 5.

[7] A. O. Adewumi and A. A. Akinyelu, ``A survey of machinelearning and nature-inspired based credit card fraud detection techniques," Int. J. Syst. Assurance Eng. Manage., vol. 8, no. 2.

[8] Suman and Nutan "Review paper on credit card fraud detection", International Journal of Computer Trends and Technology (IJCTT) – volume 4 Issue 7–July 2013.

[9] L. Delamaire, H. Abdou and J. Poinon, "Credit card fraud and detection techniques: a review", Banks and Bank Systems, Volume 4, Issue 2, 2009.

[10] Phua, D. Alahakoon and V. Lee, "Minority report in fraud detection: classification of skewed data," ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 50-59, 2004.

[11] S. Esakkiraj and S. Chidambaram, "A predictive approach for fraud detection using hidden markov model" International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 1, January- 2013