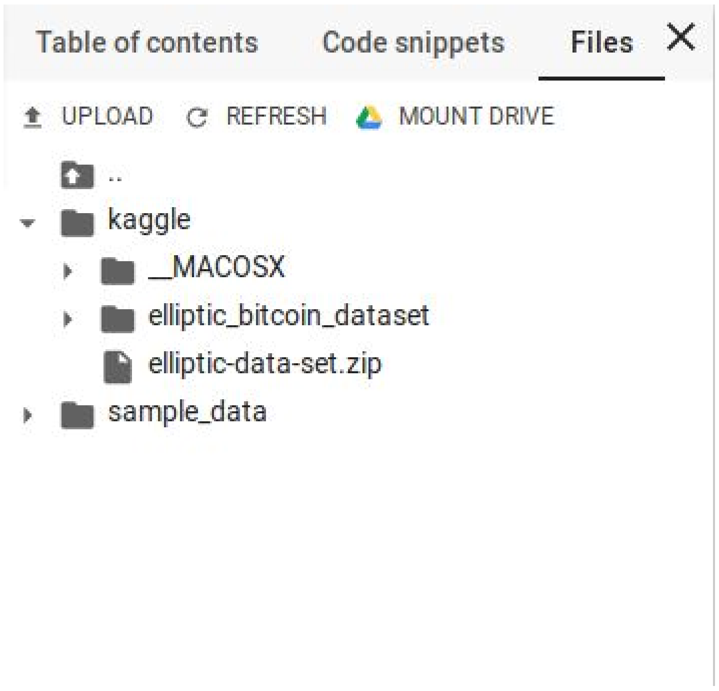
Transaction Legality Classification Task

Sayak Kundu

(This Task was done on Google Colab)

File Structure



Problem statement and short solution description.

In this task, I wanted to classify bitcoin transactions as licit or illicit. Thus I have proposed the way to detect malicious transactions. Here is the link to the data-set.

This problem was chosen because the Elliptic Data Set maps Bitcoin transactions to real entities belonging to licit categories (exchanges, wallet providers, miners, licit services, etc.) versus illicit ones (scams, malware, terrorist organizations, ransomware, Ponzi schemes, etc.). This will help us identify whether a person should continue or not with the given transaction. This prototype can be exposed as an API and can be used by cryptocurrency wallets to protect its users from malware and scams.

For this task I have used the dataset given above with 166 anonymized features collected from real transactions. Then I used binary classifiers to categorize such transactions.

EDA.

For the EDA part of the work the following things have been done. They are done in the notebook. They are as follows:

- Histogram of different classes of target

Shows counts of different class values in the dataset.

- Correlation matrix between all data features

Display pairwise correlations between all pairs of features in the data.

- Number of transactions of different types per timestamp

The plot displays how much transactions of each class were performed during each value of timestamp column.

- Histogram of the most important features according to Decision Tree Model

Visually shows importance of most important features calculated by the decision tree.

- Distribution of the most important features

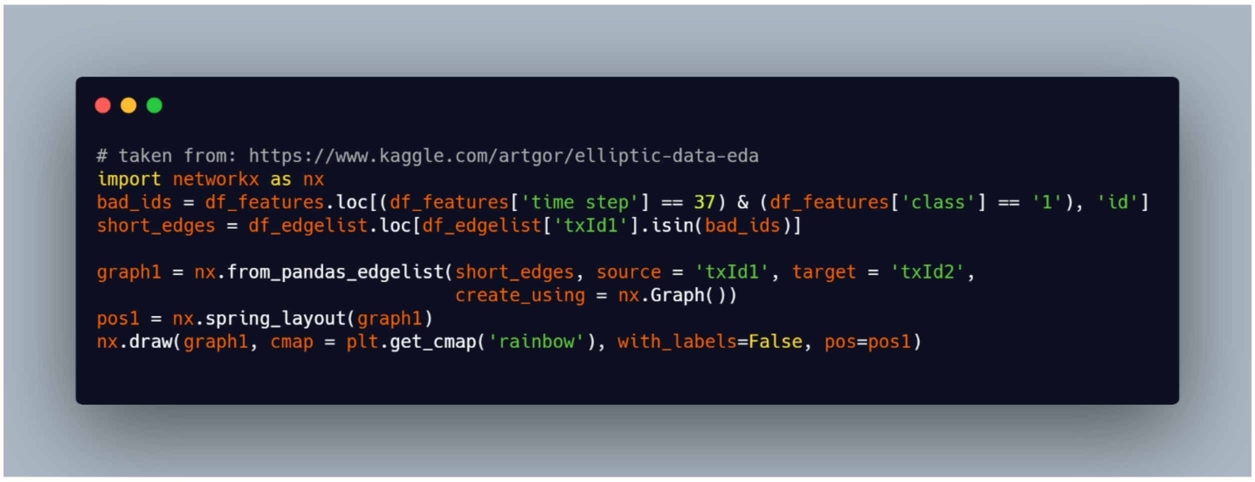
Displays distribution of each important feature according to the decision tree.

- Boxplots of the most important features

Show the mean, min, max and concentration of values for each important feature

- Connections graph identifying all groups performing malicious operations

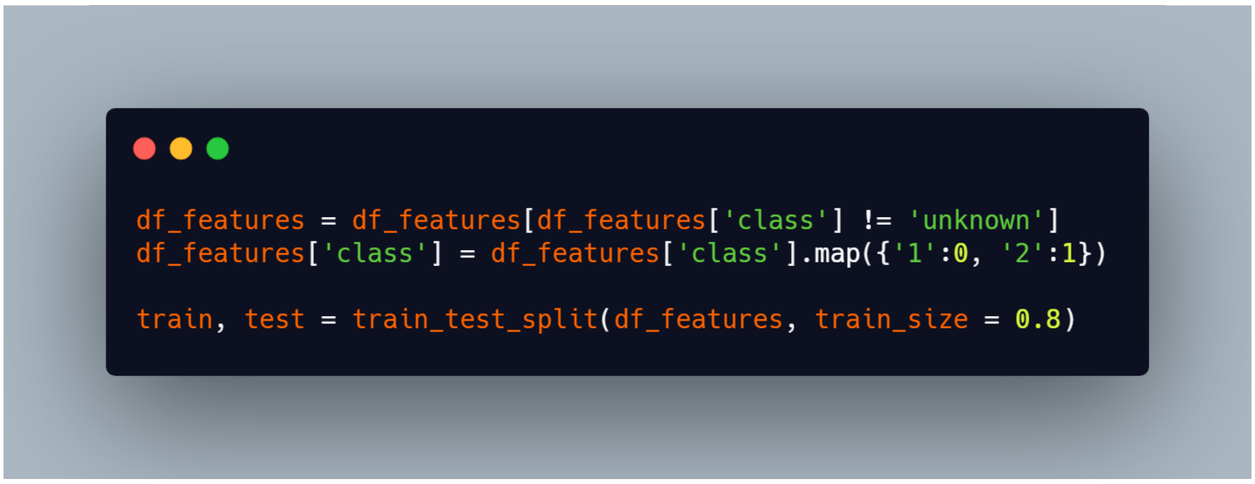
Using table df_edgelist I found transactions where first column id performed illicit operations for a certain timestamp. Then I found the targets of such operations and plotted them on the graph. From the graph it can be seen that some attackers grouped together to perform illicit operations, while others worked alone.



```python
# taken from: https://www.kaggle.com/artgor/elliptic-data-eda
import networkx as nx
bad_ids = df_features.loc[(df_features['time step'] == 37) & (df_features['class'] == '1'), 'id']
short_edges = df_edgelist.loc[df_edgelist['txId1'].isin(bad_ids)]

graph1 = nx.from_pandas_edgelist(short_edges, source = 'txId1', target = 'txId2',
                                 create_using = nx.Graph())
pos1 = nx.spring_layout(graph1)
nx.draw(graph1, cmap = plt.get_cmap('rainbow'), with_labels=False, pos=pos1)
```

Preprocessing.

I have taken only parts of the data, which belong to class 1 or 2, unknown class data has been removed from the training/testing part. Data was divided into train, test and validations parts. For labels class ‘1’ was replaced with 0, and class ‘2’ was replaced with 1.

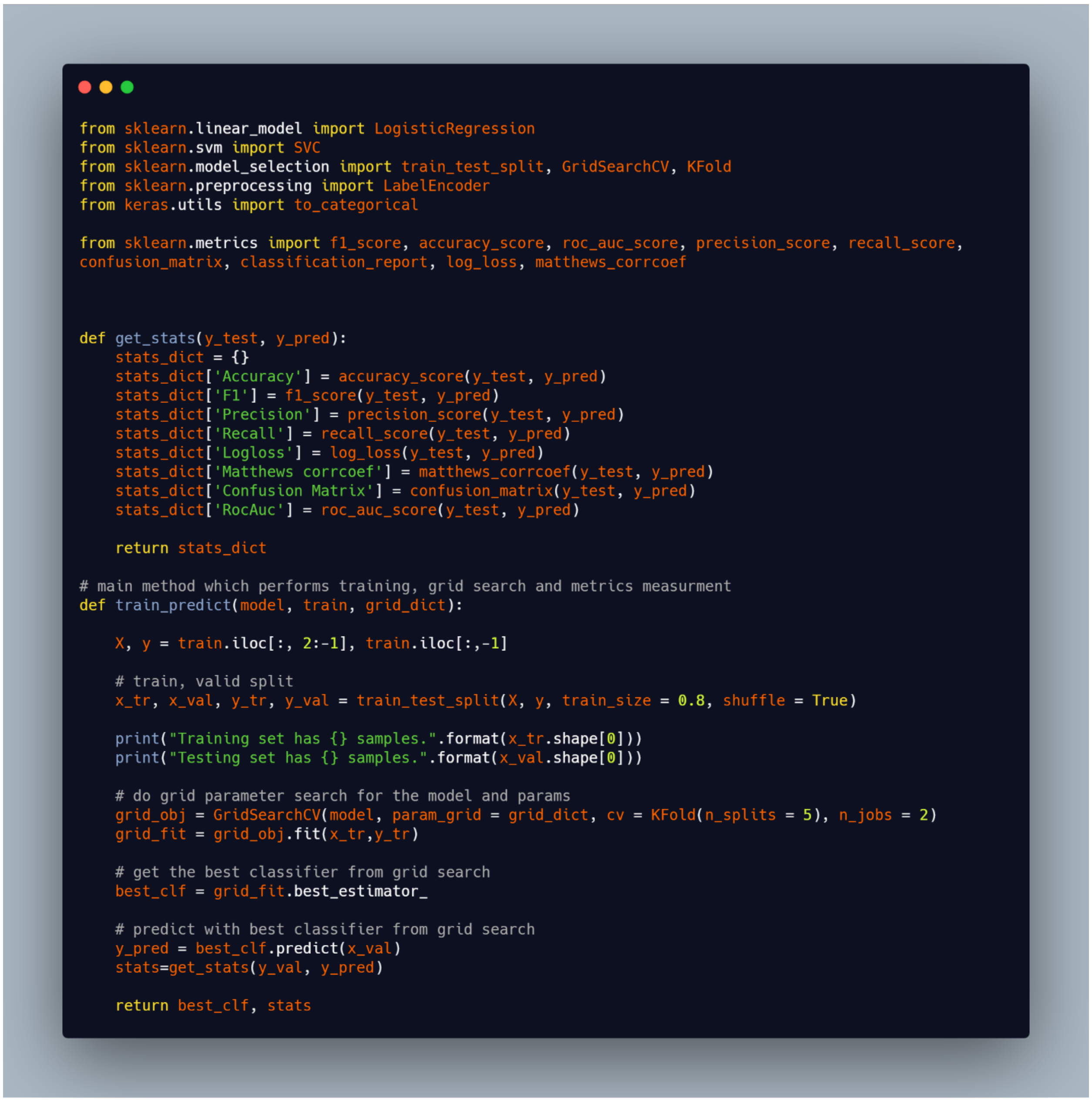


```
df_features = df_features[df_features['class'] != 'unknown']
df_features['class'] = df_features['class'].map({'1':0, '2':1})

train, test = train_test_split(df_features, train_size = 0.8)
```

Modelling

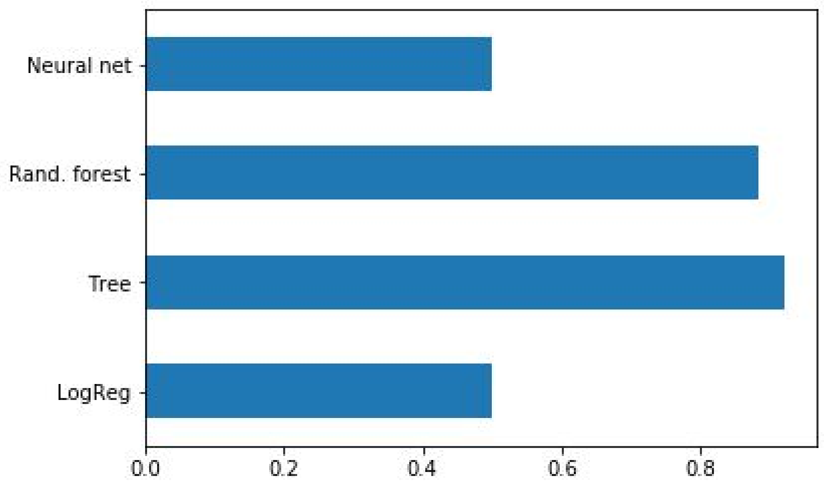
I have used 4 algorithms of different type to predict on the given data. For each algorithm (except neural network in keras) I ran grid search to identify the best set of parameters for each model.

Also KFold cross validation was done during grid search, so grid search statistics are being averaged across K-Folds. After doing grid search (simple KFold for neural network) I chose the best parameter set for each model and displayed various statistics for each best model.

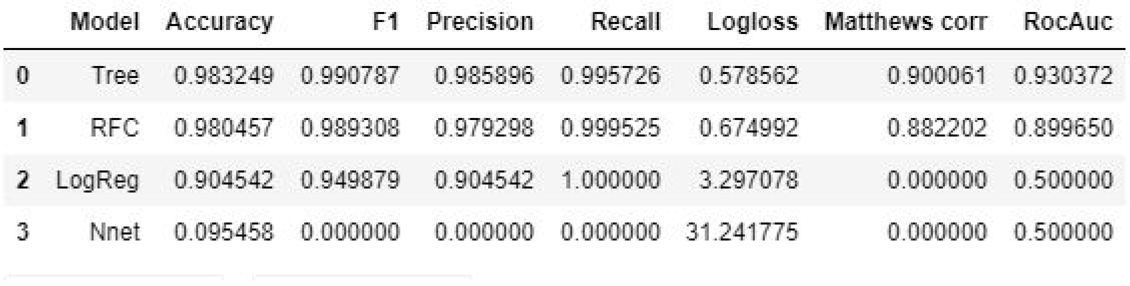


```python
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical

from sklearn.metrics import f1_score, accuracy_score, roc_auc_score, precision_score, recall_score,
confusion_matrix, classification_report, log_loss, matthews_corrcoef



def get_stats(y_test, y_pred):
    stats_dict = {}
    stats_dict['Accuracy'] = accuracy_score(y_test, y_pred)
    stats_dict['F1'] = f1_score(y_test, y_pred)
    stats_dict['Precision'] = precision_score(y_test, y_pred)
    stats_dict['Recall'] = recall_score(y_test, y_pred)
    stats_dict['Logloss'] = log_loss(y_test, y_pred)
    stats_dict['Matthews corrcoef'] = matthews_corrcoef(y_test, y_pred)
    stats_dict['Confusion Matrix'] = confusion_matrix(y_test, y_pred)
    stats_dict['RocAuc'] = roc_auc_score(y_test, y_pred)

    return stats_dict

# main method which performs training, grid search and metrics measurment
def train_predict(model, train, grid_dict):

    X, y = train.iloc[:, 2:-1], train.iloc[:,-1]

    # train, valid split
    x_tr, x_val, y_tr, y_val = train_test_split(X, y, train_size = 0.8, shuffle = True)

    print("Training set has {} samples.".format(x_tr.shape[0]))
    print("Testing set has {} samples.".format(x_val.shape[0]))

    # do grid parameter search for the model and params
    grid_obj = GridSearchCV(model, param_grid = grid_dict, cv = KFold(n_splits = 5), n_jobs = 2)
    grid_fit = grid_obj.fit(x_tr,y_tr)

    # get the best classifier from grid search
    best_clf = grid_fit.best_estimator_

    # predict with best classifier from grid search
    y_pred = best_clf.predict(x_val)
    stats=get_stats(y_val, y_pred)

    return best_clf, stats
```

During the modelling part, I used 4 model types: logistic regression, decision tree, random forest classifier and Neural Net with 3 hidden layers. During the validation and testing parts it became obvious that linear models and even neural net with more complex structure can’t detect dependencies between target and feature values, you could see that roc auc score for both Net and LogReg are terrible. Also adding more layers to Neural Network will not help either, as we don’t have sufficient amount of data, so we will only overfit to training data.

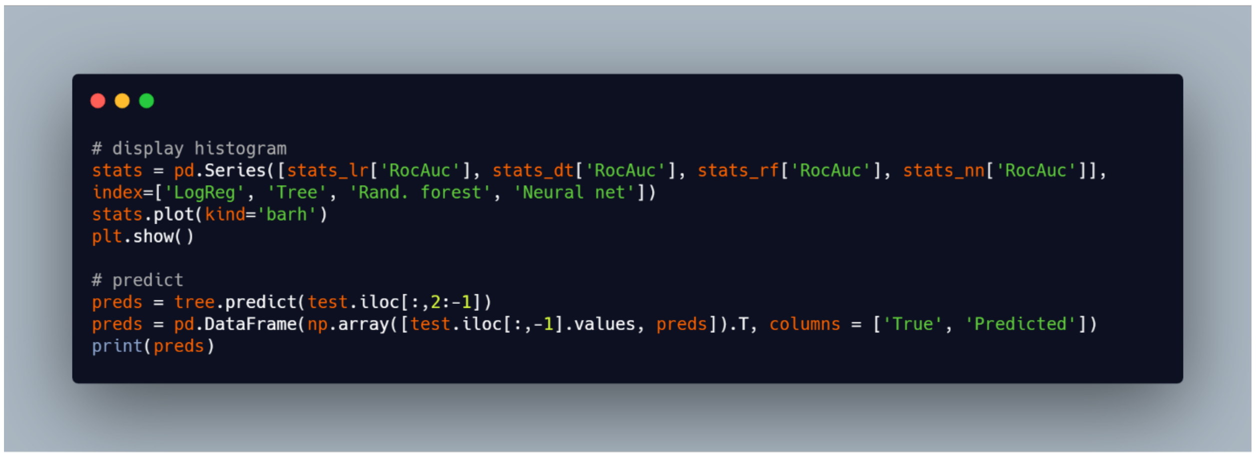


Roc Auc Scores



	Model	Accuracy	F1	Precision	Recall	Logloss	Matthews corr	RocAuc
0	Tree	0.983249	0.990787	0.985896	0.995726	0.578562	0.900061	0.930372
1	RFC	0.980457	0.989308	0.979298	0.999525	0.674992	0.882202	0.899650
2	LogReg	0.904542	0.949879	0.904542	1.000000	3.297078	0.000000	0.500000
3	Nnet	0.095458	0.000000	0.000000	0.000000	31.241775	0.000000	0.500000

Finally, I tested and predicted with the best model on the unseen test set. The final statistics for the test set are around 0.9 roc-auc score, which is quite satisficing.



```
# display histogram
stats = pd.Series([stats_lr['RocAuc'], stats_dt['RocAuc'], stats_rf['RocAuc'], stats_nn['RocAuc']],
index=['LogReg', 'Tree', 'Rand. forest', 'Neural net'])
stats.plot(kind='barh')
plt.show()

# predict
preds = tree.predict(test.iloc[:,2:-1])
preds = pd.DataFrame(np.array([test.iloc[:,-1].values, preds]).T, columns = ['True', 'Predicted'])
print(preds)
```

Libraries used.

The code was written in Python with the usage of the following libraries:

Numpy, Pandas, Sklearn, keras, tensorflow, seaborn, matplotlib.

Dataset sourse.

https://www.kaggle.com/ellipticco/elliptic-data-set

Code sources.

https://www.kaggle.com/artgor/elliptic-data-eda