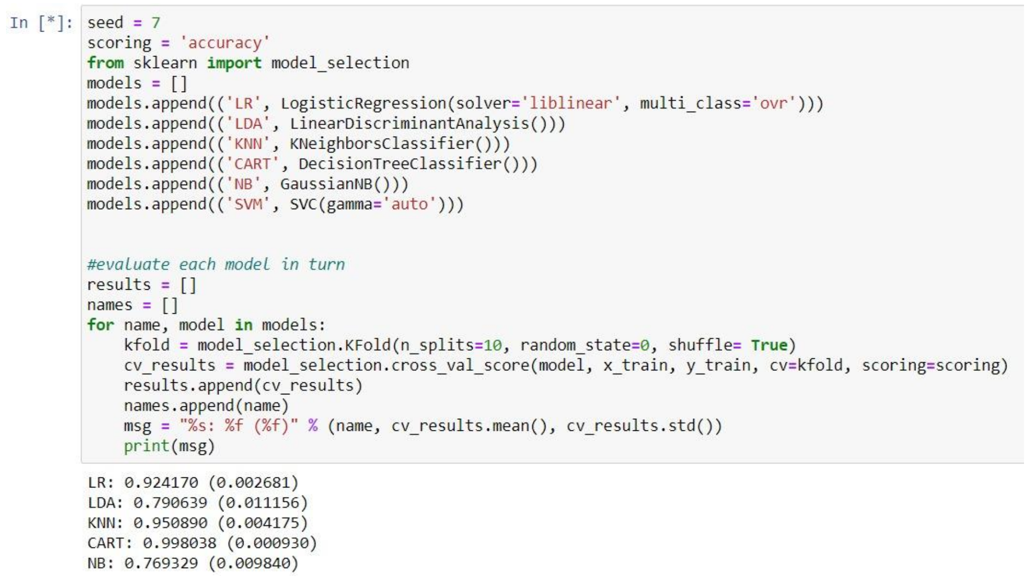
Team Runtime Terror
ML Model

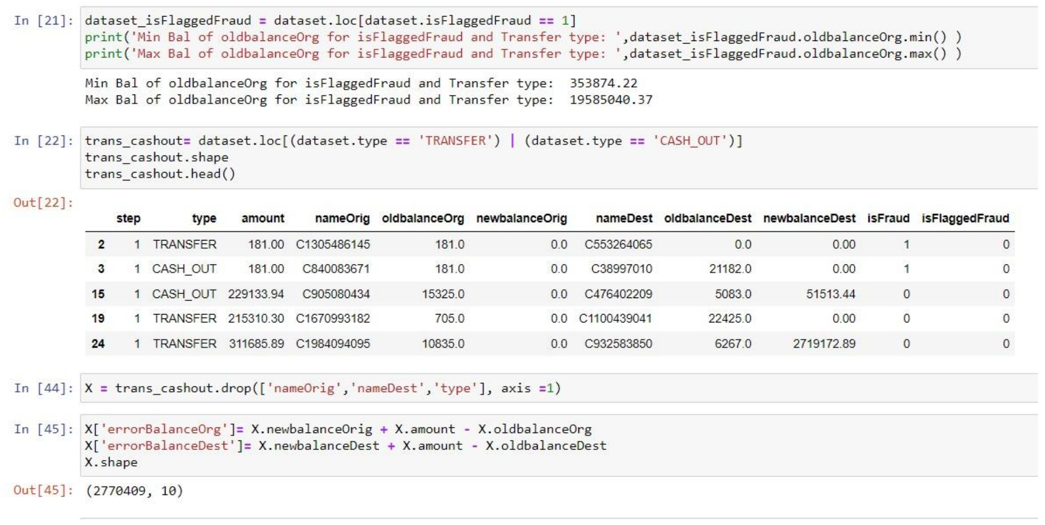
Models and their Accuracies:



```
In [*]: seed = 7
        scoring = 'accuracy'
        from sklearn import model_selection
        models = []
        models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
        models.append(('LDA', LinearDiscriminantAnalysis()))
        models.append(('KNN', KNeighborsClassifier()))
        models.append(('CART', DecisionTreeClassifier()))
        models.append(('NB', GaussianNB()))
        models.append(('SVM', SVC(gamma='auto')))


        #evaluate each model in turn
        results = []
        names = []
        for name, model in models:
            kfold = model_selection.KFold(n_splits=10, random_state=0, shuffle= True)
            cv_results = model_selection.cross_val_score(model, x_train, y_train, cv=kfold, scoring=scoring)
            results.append(cv_results)
            names.append(name)
            msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
            print(msg)

        LR: 0.924170 (0.002681)
        LDA: 0.790639 (0.011156)
        KNN: 0.950890 (0.004175)
        CART: 0.998038 (0.000930)
        NB: 0.769329 (0.009840)
```



```
In [21]: dataset_isFlaggedFraud = dataset.loc[dataset.isFlaggedFraud == 1]
         print('Min Bal of oldbalanceOrg for isFlaggedFraud and Transfer type: ',dataset_isFlaggedFraud.oldbalanceOrg.min() )
         print('Max Bal of oldbalanceOrg for isFlaggedFraud and Transfer type: ',dataset_isFlaggedFraud.oldbalanceOrg.max() )

         Min Bal of oldbalanceOrg for isFlaggedFraud and Transfer type:  353874.22
         Max Bal of oldbalanceOrg for isFlaggedFraud and Transfer type:  19585040.37

In [22]: trans_cashout= dataset.loc[(dataset.type == 'TRANSFER') | (dataset.type == 'CASH_OUT')]
         trans_cashout.shape
         trans_cashout.head()

Out[22]:
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2	1	TRANSFER	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	1	0
15	1	CASH_OUT	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	0	0
19	1	TRANSFER	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	0	0
24	1	TRANSFER	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	0	0

```
In [44]: X = trans_cashout.drop(['nameOrig','nameDest','type'], axis =1)

In [45]: X['errorBalanceOrg']= X.newbalanceOrig + X.amount - X.oldbalanceOrg
         X['errorBalanceDest']= X.newbalanceDest + X.amount - X.oldbalanceDest
         X.shape

Out[45]: (2770409, 10)
```

```
In [56]: x_train, x_test, y_train, y_test = train_test_split(X_res, y_res, test_size = 0.2, random_state = 42)
```

```
In [57]: print("Shape of x_train: ", x_train.shape)
         print("Shape of y_train: ", y_train.shape)

         print("Shape of x_test: ", x_test.shape)
         print("Shape of y_test: ", y_test.shape)
```

```
Shape of x_train:  (14783, 9)
Shape of y_train:  (14783,)
Shape of x_test:  (3696, 9)
Shape of y_test:  (3696,)
```

```
In [58]: model = LogisticRegression()
         model.fit(x_train, y_train)
         predictions= model.predict(x_test)
         print(accuracy_score(y_test, predictions))
         print(confusion_matrix(y_test, predictions))
         print(classification_report(y_test, predictions))
```

```
0.9199134199134199
[[1821  236]
 [  60 1579]]
              precision    recall  f1-score   support

           0       0.97      0.89      0.92      2057
           1       0.87      0.96      0.91      1639

    accuracy                           0.92      3696
   macro avg       0.92      0.92      0.92      3696
weighted avg       0.92      0.92      0.92      3696
```

```
In [52]: rus = RandomUnderSampler(sampling_strategy=0.8)
         X_res, y_res = rus.fit_resample(X, y)
         print(X_res.shape, y_res.shape)
         print(pd.value_counts(y_res))
```

```
(18479, 9) (18479,)
0    10266
1     8213
Name: isFraud, dtype: int64
```

```
In [56]: x_train, x_test, y_train, y_test = train_test_split(X_res, y_res, test_size = 0.2, random_state = 42)
```

```
In [57]: print("Shape of x_train: ", x_train.shape)
         print("Shape of y_train: ", y_train.shape)

         print("Shape of x_test: ", x_test.shape)
         print("Shape of y_test: ", y_test.shape)
```

```
Shape of x_train:  (14783, 9)
Shape of y_train:  (14783,)
Shape of x_test:  (3696, 9)
Shape of y_test:  (3696,)
```

```
In [73]: model = LogisticRegression()
         model.fit(x_train, y_train)
         predictions= model.predict(x_test)
         print(accuracy_score(y_test, predictions))
         print(confusion_matrix(y_test, predictions))
         print(classification_report(y_test, predictions))
```

```
0.9071969696969697
[[3821  310]
 [ 376 2885]]
              precision    recall  f1-score   support

           0       0.91      0.92      0.92      4131
           1       0.90      0.88      0.89      3261

    accuracy                           0.91      7392
   macro avg       0.91      0.90      0.91      7392
weighted avg       0.91      0.91      0.91      7392
```

In [75]:
```
cart=DecisionTreeClassifier()
cart.fit(x_train, y_train)
predictions=cart.predict(x_test)
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
0.9851190476190477
[[4069   62]
 [  48 3213]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4131
           1       0.98      0.99      0.98      3261

    accuracy                           0.99      7392
   macro avg       0.98      0.99      0.98      7392
weighted avg       0.99      0.99      0.99      7392
```