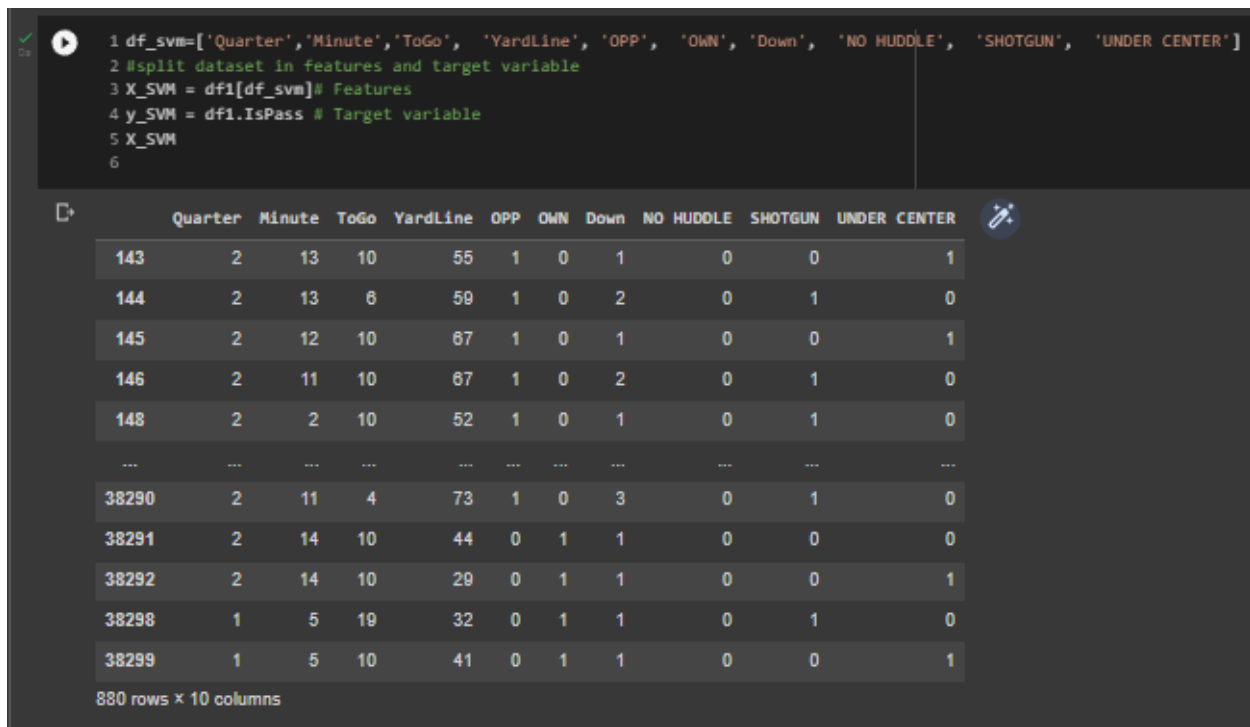In HW3 we discussed how Decision tree can predict whether a play is a run vs pass play. The base model without any tunning we were able to predict a score of 64% accuracy. We saw that Formation had a high importance in indicating whether or not the ball would be rushed. The way the team lined up, it seems, gave great predictive power for when the team would Rush, and was used to help the model discern how to tell the difference. We sew the first split is Formation (under Center) with the second split being Down < 1.5 . Next split Downs, Minute, yard line, and ToGo (Yards the Offense had left to get a first down) had the highest positive impact on our prediction capabilities. Then we created a shallow decision tree with max depth of 3 and there our model accuracy improved to 71% which was a 6 percent improvement on our baseline model.

In this HW, we will apply SVM to our classification model and check whether we get a more accurate model. Our main metric we will determine our result on is Accuracy as this is not a highly sensitive field in terms of false positive and false negatives.

We started off by using the same data set which was already cleansed, and any missing data was removed. Also any of the auto correlated features were removed. We used the same attributes in this algorithms in order to have fair comparison between decision tree and SVM.

```python
1 df_svm=['Quarter','Minute','ToGo', 'YardLine', 'OPP', 'OWN', 'Down', 'NO HUDDLE', 'SHOTGUN', 'UNDER CENTER']
2 #split dataset in features and target variable
3 X_SVM = df1[df_svm]# Features
4 y_SVM = df1.IsPass # Target variable
5 X_SVM
6
```

	Quarter	Minute	ToGo	YardLine	OPP	OWN	Down	NO HUDDLE	SHOTGUN	UNDER CENTER
143	2	13	10	55	1	0	1	0	0	1
144	2	13	8	59	1	0	2	0	1	0
145	2	12	10	67	1	0	1	0	0	1
146	2	11	10	67	1	0	2	0	1	0
148	2	2	10	52	1	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...
38290	2	11	4	73	1	0	3	0	1	0
38291	2	14	10	44	0	1	1	0	0	0
38292	2	14	10	29	0	1	1	0	0	1
38298	1	5	19	32	0	1	1	0	1	0
38299	1	5	10	41	0	1	1	0	0	1

880 rows × 10 columns

Before even doing the analysis, I figured there will not be a huge jump in accuracy because we do not have a large data set with large number of attributes. Generally, SVM are particularly effective in high-dimensional spaces and are often used in situations where the data is linearly separable. Because we only had 10 attributes in our training data, I felt as SVM would just be as good as the decision tree or worst. Compare to SVM Decision trees are typically faster to train and interpret than SVM and can handle non-linear relationships between features and target variables well.

Next, we trained an SVM model on the same training set and evaluate its performance on the same validation set. We then compared the performance of the decision tree model and the SVM model. We saw that the baseline SVM model gave us an accuracy of 71% which is 6 % better than our decision tree baseline model. The precsion was 72% and recall was 83%.

```
1 print("Accuracy:",metrics.accuracy_score(y_test_svc, y_pred_svc))
2 print("precision:", metrics.precision_score(y_test_svc, y_pred_svc))
3 #recall score
4 print("recall" , metrics.recall_score(y_test_svc, y_pred_svc))
5 print(metrics.classification_report(y_test_svc, y_pred_svc))
```

```
Accuracy: 0.7083333333333334
precision: 0.7252747252747253
recall 0.8301886792452831
              precision    recall  f1-score   support

           0       0.67      0.52      0.59       105
           1       0.73      0.83      0.77       159

    accuracy                           0.71       264
   macro avg       0.70      0.68      0.68       264
weighted avg       0.70      0.71      0.70       264
```

This was a nice baseline accuracy to start with, so the next step we took was to standardize the data and whether that helps improve the accuracy. Here after scaling the data, we were getting the same accuracy and precision. This was unusual because normally scaling and standardizing data for SVM gives better accuracy as it can compute distances standardized and can easily compare it with other distances on the same scaling. This was one which I need to further investigate why standardization did not help improve or even decrease my accuracy levels. You can see below we are obtaining the same accuracy as unscaled data.

```
[53]  1 sc= StandardScaler().fit(X_train_svc)
      2 X_train_std = sc.transform(X_train_svc)
      3 X_test_std = sc.transform(X_test_svc)
      4 svm_std = SVC(kernel='linear')
```

```
[54]  1 svm_std.fit(X_test_std,y_test_svc)
      2 y_test_pred_svc = svm_std.predict(X_test_std)
      3 y_train_pred_svc = svm_std.predict(X_train_std)
```

```
[55]  1 confusion_matrix( y_test_svc,y_test_pred_svc)

      array([[ 55,  50],
             [ 27, 132]])
```

```
[57]  1 print("Accuracy:",metrics.accuracy_score(y_test_svc, y_test_pred_svc))

      Accuracy: 0.7083333333333334
```

```
[43]  1 clf.predict([[4,13,10,55,1,1,1,1,0,5]])

      array([0])
```

The recommendation for which algorithm to use would depend on the specifics of the data and the desired trade-off between accuracy, interpretability, and computational efficiency. Here for this classification problem, I would recommend using Decision tree vs using SVM as Decision tree is easily traceable and highly interpretable. In SVM, many times you must standardize the data which may in the process lose interpretability in real application. Overall SVM models tend to be more accurate than decision tree models, but they can be slower to train and harder to interpret.