In [100]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

In [101]:
```
data = pd.read_csv(r'C:\Users\modid\Desktop\NBA_Rookie_Predictions.csv')
data.shape
#importing the dataset in and checking shape
```

Out[101]:
```
(1340, 21)
```

In [102]:
```
data_new = data.dropna()
data_new.shape
#dropping the rows with missing values
```

Out[102]:
```
(1329, 21)
```

In [103]:
```
X = data_new.drop(['name', 'target_5yrs'], axis=1)
y = data_new['target_5yrs']
#splitting the predictors and target
```

In [104]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
#splitting the data into train and test models
```

In [105]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
log_reg = LogisticRegression(max_iter = 10000)
log_reg.fit(X_train,y_train)#fit to training data
y_pred_log = log_reg.predict(X_test)#predict on test data
log_cm = confusion_matrix(y_test, y_pred_log)
log_cm
```

Out[105]:
```
array([[ 84,  69],
       [ 44, 202]], dtype=int64)
```

In [106]:
```
error_rate = 1 - accuracy_score(y_test, y_pred_log)
error_rate
#obtaining accuracy of model
```

Out[106]:
```
0.28320802005012535
```

In [107]:
```
coefficients = log_reg.coef_[0]
coefficients
#obtaining coefficients
```

Out[107]:
```
array([ 0.03608031, -0.04147289,  0.14324291, -0.72781581,  0.24059492,
        0.05091288,  1.44149439, -0.56181202,  0.00495857,  0.06826616,
       -0.05954725,  0.01208377,  0.58801621, -0.27307952,  0.11206921,
        0.18578501, -0.06734523,  0.75187748,  0.06231129])
```

In [108]:
```
log_rep = classification_report(y_test, y_pred_log)
log_rep
#goodness of fit
```

Out[108]:
```
'              precision    recall  f1-score   support\n\n           0       0.66      0.55      0.60
153\n           1       0.75      0.82      0.78       246\n\n    accuracy
0.72       399\n   macro avg       0.70      0.69      0.69       399\nweighted avg       0.71
0.72      0.71       399\n'
```

In [109]:
```
std = np.std(X_train, axis=0)
sample_size = X_train.shape[0]
se = std/np.sqrt(sample_size)
se
#standard error of each of the coefficents
```

Out[109]:
```
gp         0.570715
min        0.272810
pts        0.143162
fgm        0.055426
fga        0.119436
fg         0.202989
3p_made    0.012579
3pa        0.035041
3p         0.516587
ftm        0.031636
fta        0.042296
ft         0.337288
oreb       0.024789
dreb       0.043053
reb        0.065175
ast        0.046810
stl        0.013413
blk        0.014200
tov        0.024069
dtype: float64
```

In [110]:
```
from sklearn.model_selection import cross_val_score
cv_acc = cross_val_score(log_reg, X_train, y_train, cv=5, scoring='accuracy')
cv_acc
#running cross validation on the model, using the X and y training data
#doing k fold cross validation, using 5 folds
```

Out[110]:
```
array([0.67204301, 0.67741935, 0.75268817, 0.70430108, 0.73655914])
```

In [111]:
```
average_acc = np.mean(cv_acc)
average_acc
#taking the average of all of folds to obtain how well the model performs
```

Out[111]:
```
0.7086021505376344
```

In [ ]:
```

```

In [ ]:
```

```

In [112]:
```
###LDA Analysis
```

In [113]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda_model = LinearDiscriminantAnalysis()
lda_model.fit(X_train,y_train)
y_pred_lda = lda_model.predict(X_test)
lda_cm = confusion_matrix(y_test, y_pred_lda)
lda_cm
```

Out[113]:
```
array([[ 85,  68],
       [ 43, 203]], dtype=int64)
```

In [114]:
```
error_rate_lda = 1 - accuracy_score(y_test, y_pred_lda)
error_rate_lda
```

Out[114]:
```
0.27819548872180455
```

```
In [115]: coefficients_lda = lda_model.coef_[0]
          coefficients_lda

Out[115]: array([ 4.12045874e-02, -3.26601457e-02,  1.13678743e-01, -1.45808228e+00,
                  6.09499561e-01,  8.21856565e-02,  4.01883026e+00, -1.55291668e+00,
                  2.88307629e-03, -2.09158433e-01,  1.22587172e-01,  1.70911894e-02,
                  1.25150723e+00,  4.23120527e-01, -6.06899357e-01,  1.64622230e-01,
                 -1.18019383e-02,  6.02904733e-01,  4.58506190e-02])
```

```
In [116]: lda_rep = classification_report(y_test, y_pred_lda)
          lda_rep

Out[116]: '              precision    recall  f1-score   support\n\n           0       0.66      0.56      0.60
          153\n           1       0.75      0.83      0.79       246\n\n    accuracy
          0.72       399\n   macro avg       0.71      0.69      0.70       399\nweighted avg       0.72
          0.72      0.72       399\n'
```

```
In [117]: cv_acc_lda = cross_val_score(lda_model, X_train, y_train, cv=5, scoring='accuracy')
          cv_acc_lda

Out[117]: array([0.66129032, 0.66129032, 0.75268817, 0.6827957 , 0.73655914])
```

```
In [118]: average_acc_lda = np.mean(cv_acc_lda)
          average_acc_lda

Out[118]: 0.6989247311827956
```

```
In [119]: from sklearn.utils import resample
          num_bootstraps = 1000
          coef_list = []
```

```
In [158]: for i in range(num_bootstraps):
              X_sample, y_sample = resample(X, y)
              lda = LinearDiscriminantAnalysis()
              lda.fit(X_sample, y_sample)
              coef_list.append(lda_model.coef_[0])

          coef_array = np.array(coef_list)
          coef_array

Out[158]: array([[ 0.03464551, -0.02527599, -0.52279653, ..., -0.30477843,
                   0.41237472,  0.11052142],
                 [ 0.04698796, -0.0896188 ,  0.28757558, ...,  0.27664974,
                   0.34485008, -0.30430118],
                 [ 0.04301429, -0.00233044,  0.11774165, ..., -0.39032792,
                  -0.16847388, -0.15700336],
                 ...,
                 [ 0.04120459, -0.03266015,  0.11367874, ..., -0.01180194,
                   0.60290473,  0.04585062],
                 [ 0.04120459, -0.03266015,  0.11367874, ..., -0.01180194,
                   0.60290473,  0.04585062],
                 [ 0.04120459, -0.03266015,  0.11367874, ..., -0.01180194,
                   0.60290473,  0.04585062]])
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [28]: ###QDA(Used to be K Nearest Neighbors, but issue with new Sklearn library, KNN results on my document)
```

```
In [121]: from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
          qda_model = QuadraticDiscriminantAnalysis()
          qda_model.fit(X_train,y_train)
          y_pred_qda = qda_model.predict(X_test)
          qda_cm = confusion_matrix(y_test, y_pred_qda)
          qda_cm

Out[121]: array([[116,  37],
                 [100, 146]], dtype=int64)
```

```
In [122]: error_rate_qda = 1 - accuracy_score(y_test, y_pred_qda)
          error_rate_qda

Out[122]: 0.343358395989975
```

```
In [ ]:
```

```
In [123]: qda_rep = classification_report(y_test, y_pred_qda)
          qda_rep

Out[123]: '              precision    recall  f1-score   support\n\n           0       0.54      0.76      0.63
          153\n           1       0.80      0.59      0.68       246\n\n    accuracy
          0.66       399\n   macro avg       0.67      0.68      0.65       399\nweighted avg       0.70
          0.66      0.66       399\n'
```

```
In [124]: cv_acc_qda = cross_val_score(qda_model, X_train, y_train, cv=5, scoring='accuracy')
          cv_acc_qda

Out[124]: array([0.62903226, 0.62903226, 0.69892473, 0.60752688, 0.6344086 ])
```

```
In [125]: average_acc_qda = np.mean(cv_acc_qda)
          average_acc_qda

Out[125]: 0.6397849462365591
```

```
In [ ]:
```

```
In [126]: ###Random Forest
```

```
In [127]: from sklearn.ensemble import RandomForestClassifier
          rf_model = RandomForestClassifier()
          rf_model.fit(X_train, y_train)
          y_pred_rf = rf_model.predict(X_test)
          rf_cm = confusion_matrix(y_test, y_pred_rf)
          rf_cm

Out[127]: array([[ 73,  80],
                 [ 54, 192]], dtype=int64)
```

```
In [128]: error_rate_rf = 1 - accuracy_score(y_test, y_pred_rf)
          error_rate_rf

Out[128]: 0.3358395989974937
```

In [129]:
```
rf_rep = classification_report(y_test, y_pred_rf)
rf_rep
```

Out[129]:
```
'              precision    recall  f1-score   support\n\n           0       0.57      0.48      0.52
153\n           1       0.71      0.78      0.74       246\n\n    accuracy
0.66       399\n   macro avg       0.64      0.63      0.63       399\nweighted avg       0.66
0.66      0.66       399\n'
```

In [130]:
```
cv_acc_rf = cross_val_score(rf_model, X_train, y_train, cv=5, scoring='accuracy')
cv_acc_rf
```

Out[130]:
```
array([0.67741935, 0.70430108, 0.74193548, 0.64516129, 0.70430108])
```

In [131]:
```
average_acc_rf = np.mean(cv_acc_rf)
average_acc_rf
```

Out[131]:
```
0.6946236559139786
```

In [132]:
```
feature_importance = pd.DataFrame({
    'Feature': X_train.columns,
    'Coefficient': rf_model.feature_importances_})

feature_importance
```

Out[132]:

	Feature	Coefficient
0	gp	0.111924
1	min	0.064362
2	pts	0.062196
3	fgm	0.062725
4	fga	0.050448
5	fg	0.082833
6	3p_made	0.022418
7	3pa	0.032614
8	3p	0.044514
9	ftm	0.050234
10	fta	0.052687
11	ft	0.061918
12	oreb	0.044187
13	dreb	0.047242
14	reb	0.051974
15	ast	0.046681
16	stl	0.032952
17	blk	0.038293
18	tov	0.039800

In [133]:
```
num_bootstraps = 10
feature_importances = []
```

In [134]:
```
for i in range(num_bootstraps):
    X_sample, y_sample = resample(X, y)
    rf = RandomForestClassifier()
    rf.fit(X_sample, y_sample)
    feature_importances.append(rf.feature_importances_)
```

In [135]:
```
feature_array = np.array(feature_importances)
standard_errors = np.std(feature_array, axis=0)
standard_errors
```

Out[135]:
```
array([0.01497074, 0.00657489, 0.00511298, 0.00616035, 0.00391987,
       0.00774345, 0.00174248, 0.00199712, 0.00321969, 0.00776175,
       0.00451264, 0.00460588, 0.00613606, 0.00413104, 0.00363989,
       0.00176143, 0.00234078, 0.00443688, 0.00261307])
```

In [ ]:

In [ ]:

In [136]:
```
###Lasso Regression for Classification
```

In [ ]:

In [137]:
```
from sklearn.linear_model import LogisticRegressionCV
lasso = LogisticRegressionCV(cv=5, penalty='l1', solver='liblinear', max_iter = 100000)##automatically
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)
lasso_cm = confusion_matrix(y_test, y_pred_lasso)
lasso_cm
```

Out[137]:
```
array([[ 85,  68],
       [ 48, 198]], dtype=int64)
```

In [138]:
```
error_rate_lasso = 1 - accuracy_score(y_test, y_pred_lasso)
error_rate_lasso
```

Out[138]:
```
0.2907268170426065
```

In [139]:
```
lasso_rep = classification_report(y_test, y_pred_lasso)
lasso_rep
```

Out[139]:
```
'              precision    recall  f1-score   support\n\n           0       0.64      0.56      0.59
153\n           1       0.74      0.80      0.77       246\n\n    accuracy
0.71       399\n   macro avg       0.69      0.68      0.68       399\nweighted avg       0.70
0.71      0.70       399\n'
```

In [140]:
```
coefficients_lasso = lasso.coef_[0]
coefficients_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Coefficient': coefficients_lasso
})

coefficients_df
```

Out[140]:

	Feature	Coefficient
0	gp	0.036056
1	min	-0.048045
2	pts	0.000000
3	fgm	-0.555065
4	fga	0.313201
5	fg	0.044278
6	3p_made	3.508661
7	3pa	-1.265343
8	3p	0.002964
9	ftm	0.141796
10	fta	0.000000
11	ft	0.011798
12	oreb	0.709830
13	dreb	-0.159098
14	reb	0.000000
15	ast	0.212925
16	stl	0.000000
17	blk	0.799254
18	tov	0.000000

In [141]:
```
cv_acc_lasso = cross_val_score(lasso, X_train, y_train, cv=5, scoring='accuracy')
cv_acc_lasso
```

Out[141]:
```
array([0.66666667, 0.64516129, 0.75268817, 0.67741935, 0.73655914])
```

In [142]:
```
average_acc_lasso = np.mean(cv_acc_lasso)
average_acc_lasso
```

Out[142]:
```
0.6956989247311828
```

In [143]:
```
num_bootstraps = 5
bootstrapped_coefficients = []
```

In [144]:
```
for i in range(num_bootstraps):
    X_sample, y_sample = resample(X_train, y_train)
    lasso = LogisticRegressionCV(cv=5, penalty='l1', solver='liblinear', max_iter=1000000)
    lasso.fit(X_sample, y_sample)
    bootstrapped_coefficients.append(lasso.coef_[0])
```

In [145]:
```
coefficients_array = np.array(bootstrapped_coefficients)
sd = np.std(coefficients_array, axis=0)
se = sd / np.sqrt(X_train.shape[0])
```

In [146]:
```
standard_errors
```

Out[146]:
```
array([0.01497074, 0.00657489, 0.00511298, 0.00616035, 0.00391987,
       0.00774345, 0.00174248, 0.00199712, 0.00321969, 0.00776175,
       0.00451264, 0.00460588, 0.00613606, 0.00413104, 0.00363989,
       0.00176143, 0.00234078, 0.00443688, 0.00261307])
```

In [ ]:
```

```

In [ ]:
```

```

In [147]:
```
###PCA
```

In [148]:
```
from sklearn.decomposition import PCA
```

In [149]:
```
pca = PCA(n_components=3)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

In [150]:
```
classifier = LogisticRegression()
classifier.fit(X_train_pca, y_train)
```

Out[150]:
```
▾ LogisticRegression

LogisticRegression()
```

In [151]:
```
y_pred_pca= classifier.predict(X_test_pca)
```

In [152]:
```
pca_cm = confusion_matrix(y_test, y_pred_pca)
pca_cm
```

Out[152]:
```
array([[ 79,  74],
       [ 46, 200]], dtype=int64)
```

In [153]:
```
error_rate_PCA = 1 - accuracy_score(y_test, y_pred_pca)
error_rate_PCA
```

Out[153]:
```
0.3007518796992481
```

In [154]:
```
pca_rep = classification_report(y_test, y_pred_pca)
pca_rep
```

Out[154]:
```
'               precision    recall  f1-score   support\n\n           0       0.63      0.52      0.57
153\n           1       0.73      0.81      0.77       246\n\n    accuracy
0.70       399\n   macro avg       0.68      0.66      0.67       399\nweighted avg       0.69
0.70      0.69       399\n'
```

In [155]:
```
coefficients_pca = classifier.coef_[0]
coefficients_pca
```

Out[155]:
```
array([-0.04577693, -0.02199835,  0.00417098])
```

In [156]:
```
cv_acc_pca = cross_val_score(classifier, X_train_pca, y_train, cv=5, scoring='accuracy')
cv_acc_pca
```

Out[156]:
```
array([0.69892473, 0.66666667, 0.72043011, 0.66666667, 0.7311828 ])
```

In [157]:
```
average_acc_pca = np.mean(cv_acc_pca)
average_acc_pca
```

Out[157]:
```
0.696774193548387
```

In [ ]:

In [ ]:

In [ ]:

In [56]:

```
Requirement already satisfied: scikit-learn in c:\users\modid\anaconda3\lib\site-packages (1.3.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\modid\anaconda3\lib\site-packages (from scik
it-learn) (1.3.2)
Requirement already satisfied: scipy>=1.5.0 in c:\users\modid\anaconda3\lib\site-packages (from sciki
t-learn) (1.10.1)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\modid\anaconda3\lib\site-packages (from
scikit-learn) (1.24.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\modid\anaconda3\lib\site-packages (fr
om scikit-learn) (2.1.0)
```

In [57]:
```
###Polynomial Regression, I was able to figure it out but did include on analysis.
```

In [58]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
poly_reg = PolynomialFeatures(degree = 3)
X_train_poly = poly_reg.fit_transform(X_train)
X_test_poly = poly_reg.transform(X_test)
log_poly = LogisticRegression(solver='liblinear', max_iter=10000)
log_poly.fit(X_train_poly, y_train)
y_pred_poly = log_poly.predict(X_test_poly)
poly_cm = confusion_matrix(y_test, y_pred_poly)
poly_cm
```

Out[58]:
```
array([[ 78,  81],
       [ 64, 176]], dtype=int64)
```

In [59]:
```
error_rate_poly = 1 - accuracy_score(y_test, y_pred_poly)
error_rate_poly
```

Out[59]:
```
0.3634085213032582
```

In [60]:
```
poly_rep = classification_report(y_test, y_pred_poly)
poly_rep
```

Out[60]:
```
'              precision    recall  f1-score   support\n\n           0       0.55      0.49      0.52
159\n           1       0.68      0.73      0.71       240\n\n    accuracy
0.64       399\n   macro avg       0.62      0.61      0.61       399\nweighted avg       0.63
0.64      0.63       399\n'
```

In [61]:
```
coefficients_poly = log_poly.coef_[0]
coefficients_poly
```

Out[61]:
```
array([-3.93924441e-05, -4.27216482e-04, -3.53518923e-04, ...,
        1.68082157e-04,  1.29783772e-04, -1.72661155e-04])
```

In [62]:
```
cv_acc_poly = cross_val_score(log_poly, X_train_poly, y_train, cv=5, scoring='accuracy')
cv_acc_poly
```

Out[62]:
```
array([0.60752688, 0.61827957, 0.61827957, 0.69354839, 0.6344086 ])
```

In [63]:

```
average_acc_lasso = np.mean(cv_acc_lasso)
average_acc_lasso
```

Out[63]:

```
0.7129032258064518
```

In [ ]:

In [ ]: