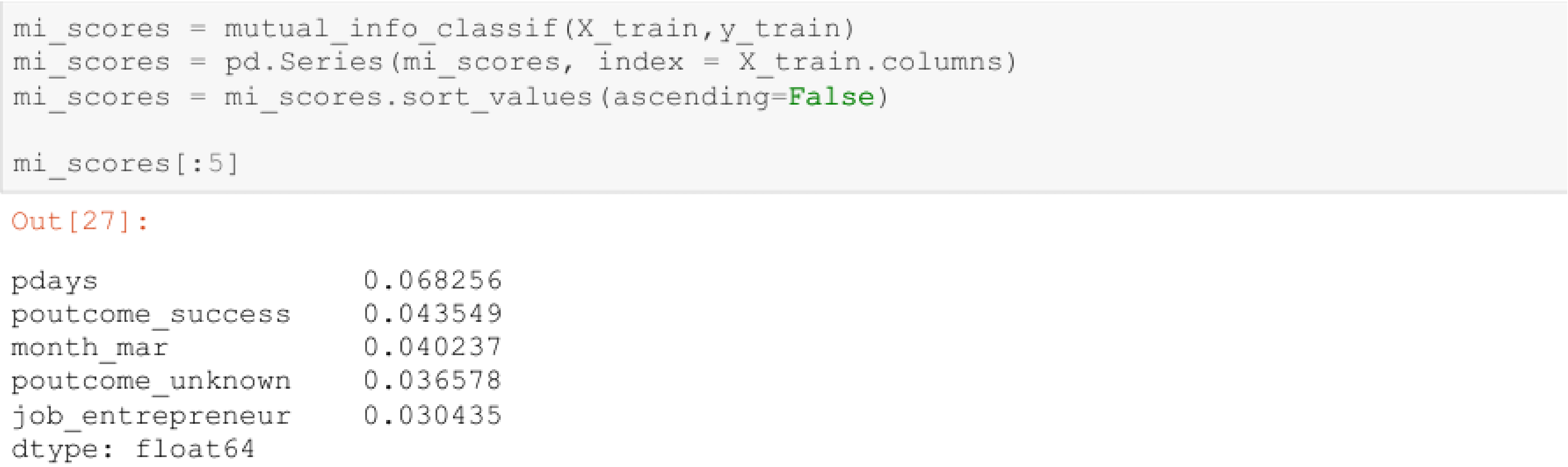
Efficient Telemarketing

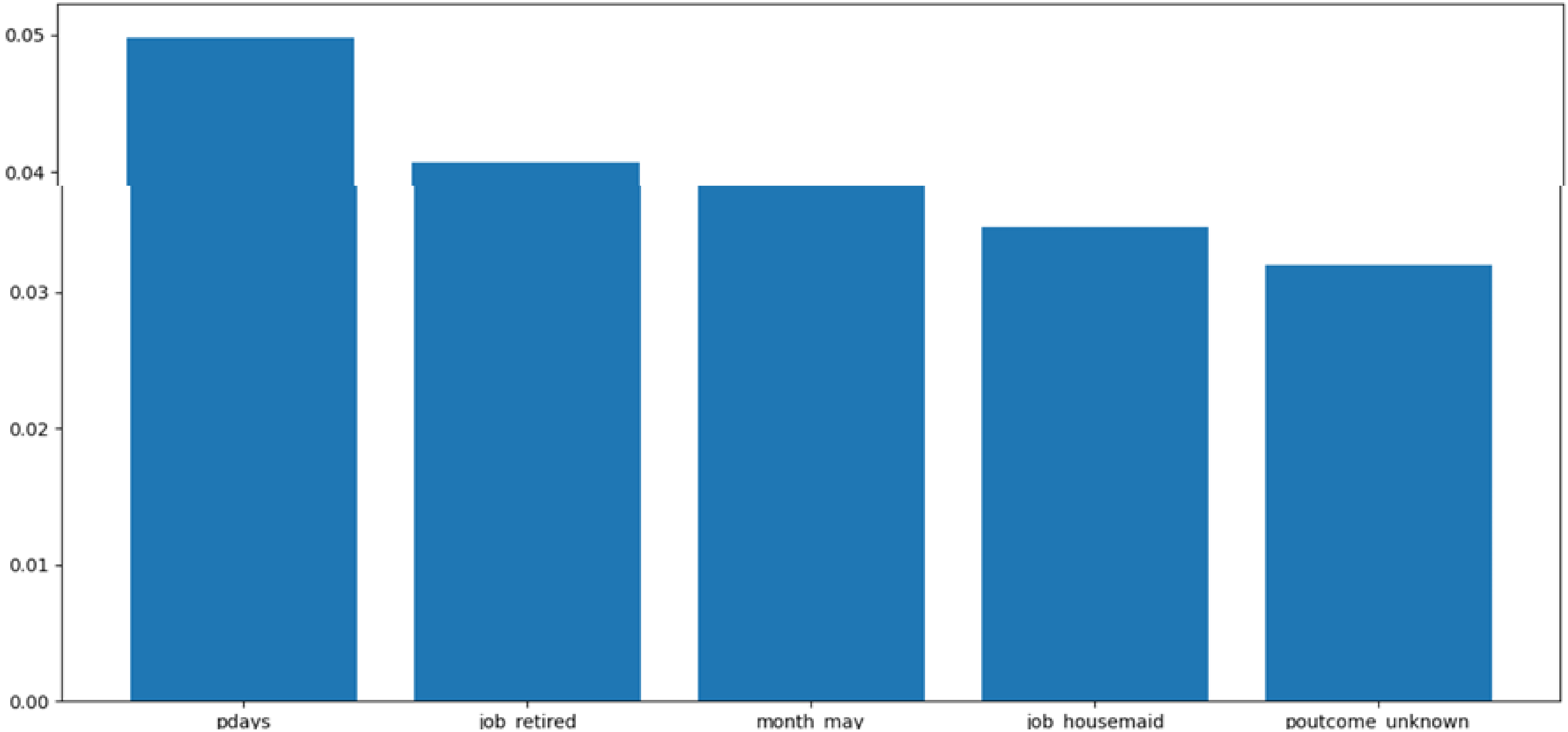


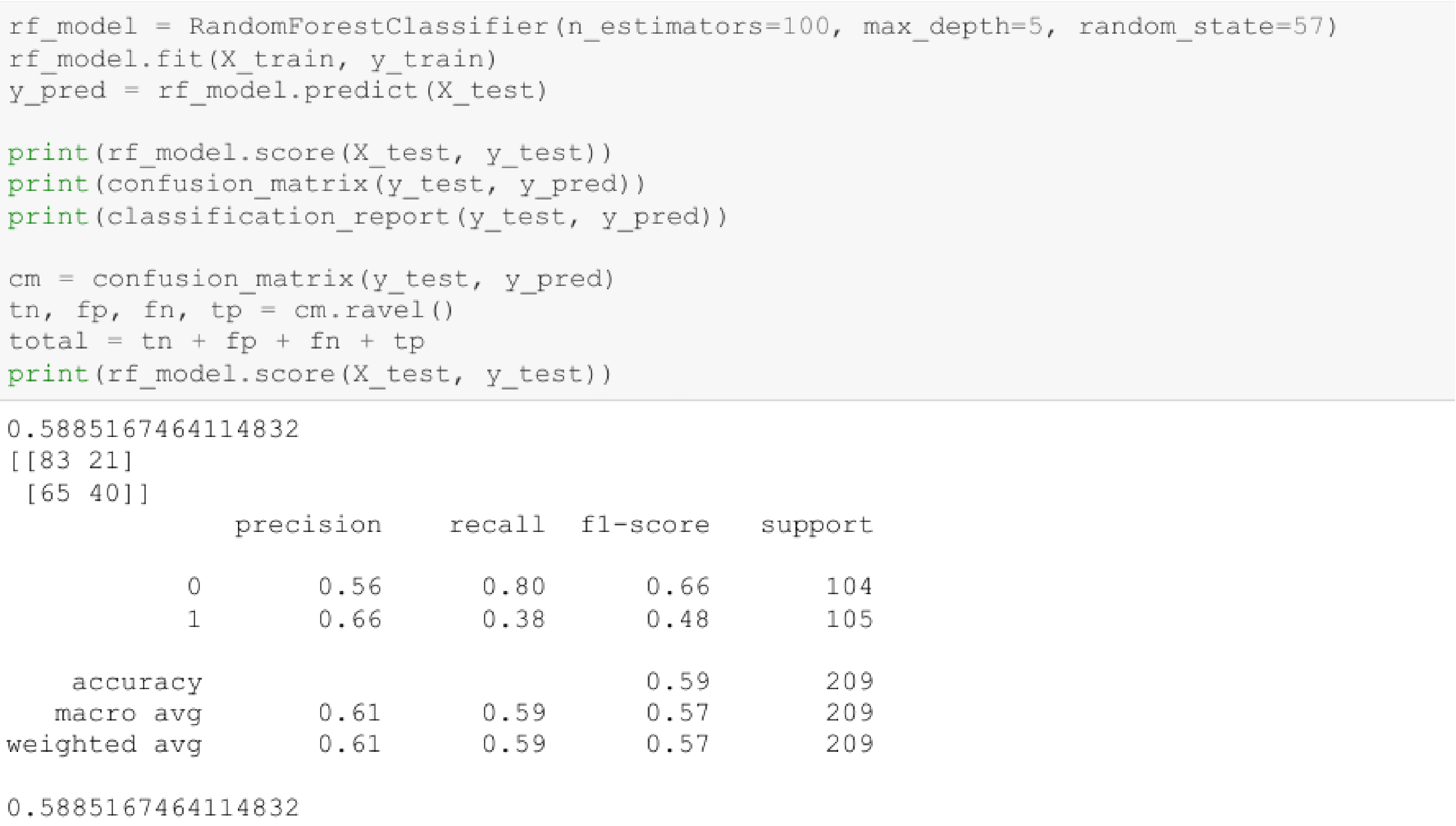
```
mi_scores = mutual_info_classif(X_train,y_train)
mi_scores = pd.Series(mi_scores, index = X_train.columns)
mi_scores = mi_scores.sort_values(ascending=False)

mi_scores[:5]
```

Out[27]:

```
pdays                 0.068256
poutcome_success      0.043549
month_mar             0.040237
poutcome_unknown      0.036578
job_entrepreneur      0.030435
dtype: float64
```

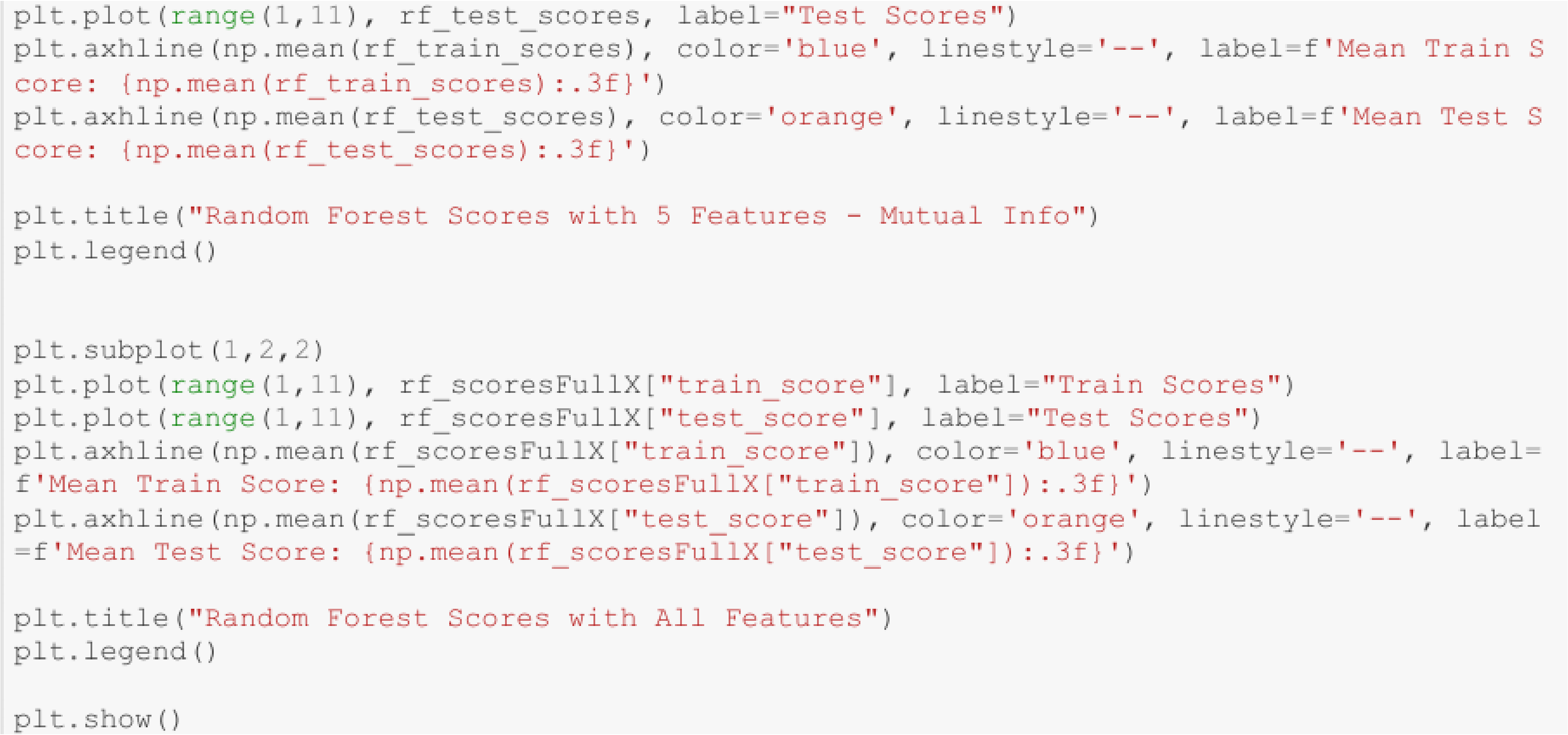
Mutual Info Classif



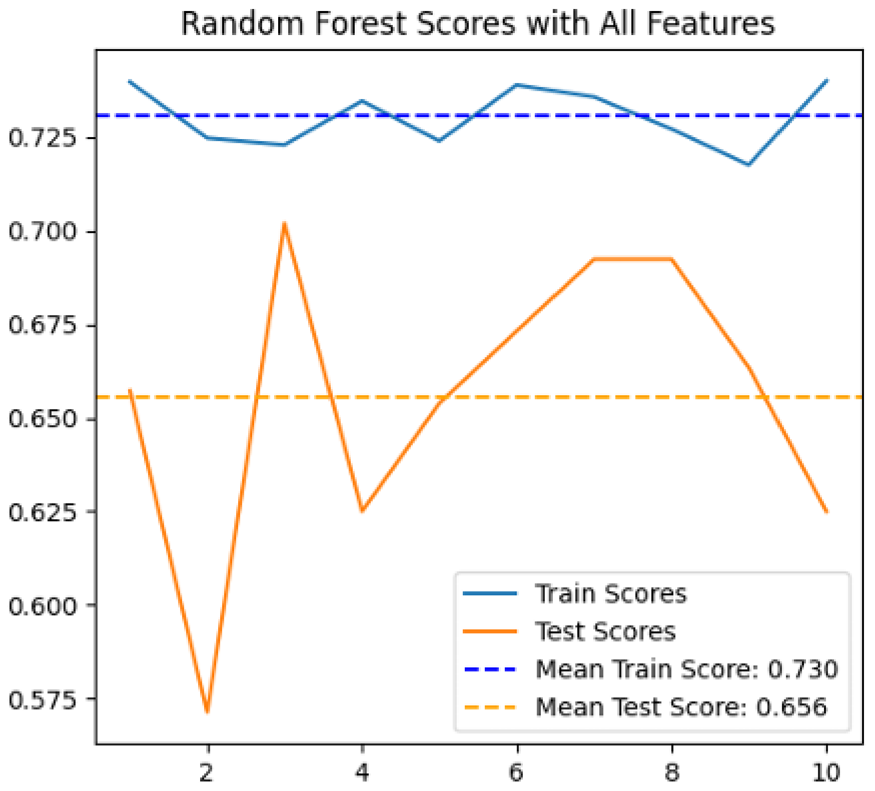


```python
rf_model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=57)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)

print(rf_model.score(X_test, y_test))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()
total = tn + fp + fn + tp
print(rf_model.score(X_test, y_test))
```

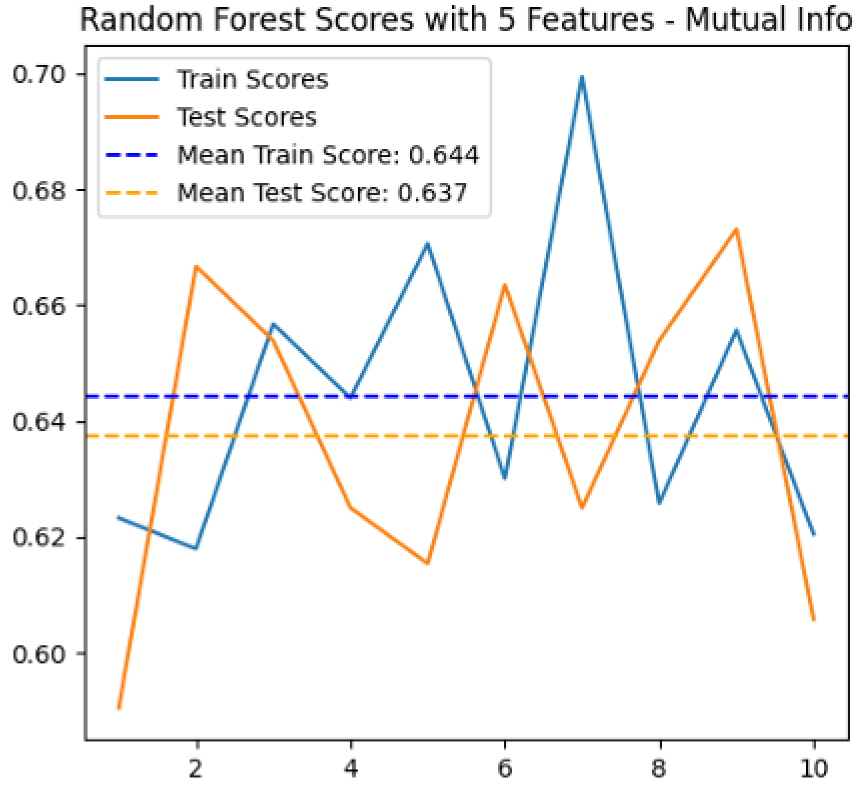
```
0.5885167464114832
[[83 21]
 [65 40]]
              precision    recall  f1-score   support

           0       0.56      0.80      0.66       104
           1       0.66      0.38      0.48       105

    accuracy                           0.59       209
   macro avg       0.61      0.59      0.57       209
weighted avg       0.61      0.59      0.57       209

0.5885167464114832
```



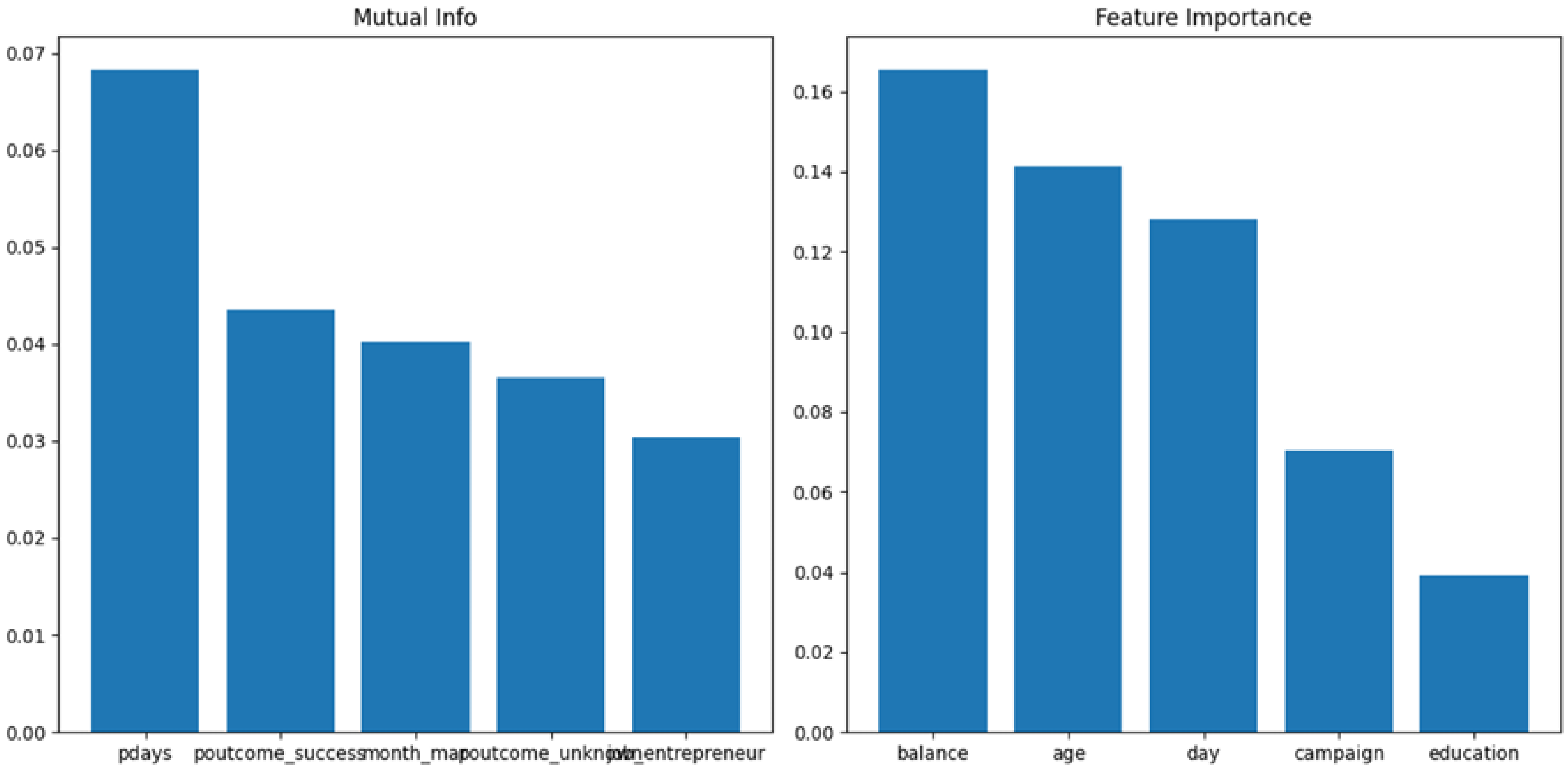
```python
plt.plot(range(1,11), rf_test_scores, label="Test Scores")
plt.axhline(np.mean(rf_train_scores), color='blue', linestyle='--', label=f'Mean Train Score: {np.mean(rf_train_scores):.3f}')
plt.axhline(np.mean(rf_test_scores), color='orange', linestyle='--', label=f'Mean Test Score: {np.mean(rf_test_scores):.3f}')

plt.title("Random Forest Scores with 5 Features - Mutual Info")
plt.legend()


plt.subplot(1,2,2)
plt.plot(range(1,11), rf_scoresFullX["train_score"], label="Train Scores")
plt.plot(range(1,11), rf_scoresFullX["test_score"], label="Test Scores")
plt.axhline(np.mean(rf_scoresFullX["train_score"]), color='blue', linestyle='--', label=f'Mean Train Score: {np.mean(rf_scoresFullX["train_score"]):.3f}')
plt.axhline(np.mean(rf_scoresFullX["test_score"]), color='orange', linestyle='--', label=f'Mean Test Score: {np.mean(rf_scoresFullX["test_score"]):.3f}')

plt.title("Random Forest Scores with All Features")
plt.legend()

plt.show()
```



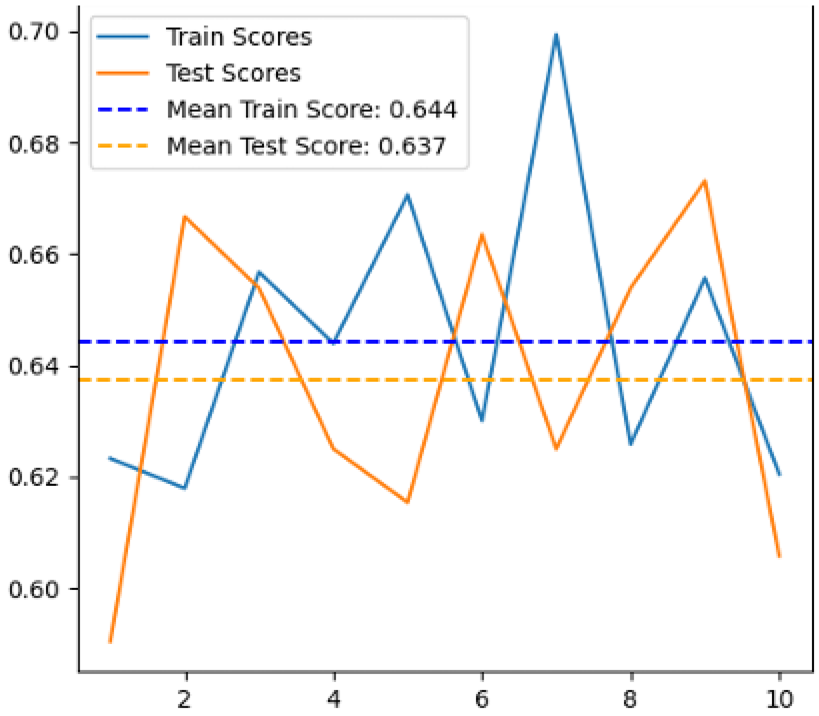
Random Forest Scores with All Features



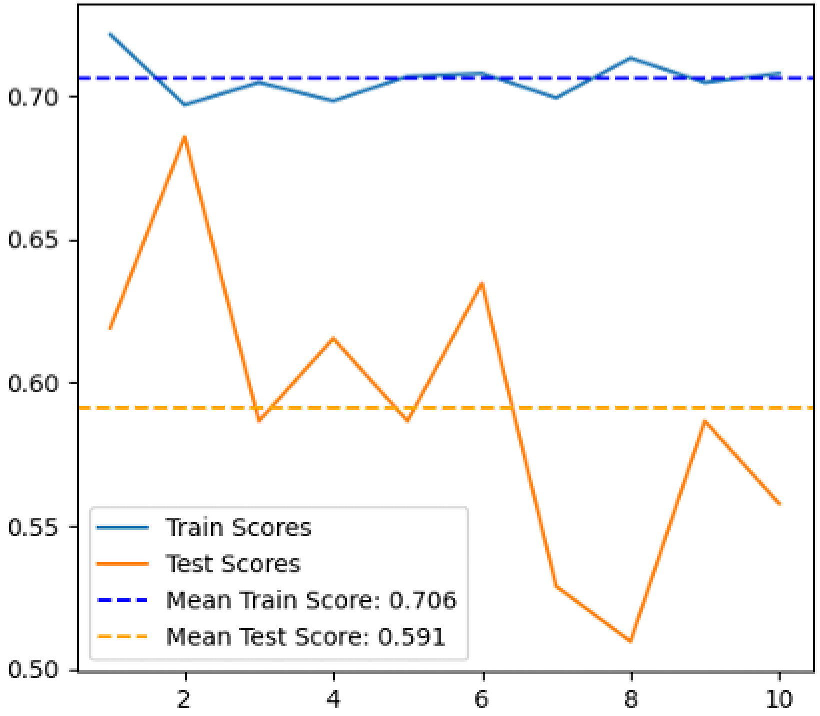
Random Forest Scores with 5 Features - Mutual Info


Mutual Info
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
pdays
poutcome_success
month_mar
poutcome_unknown
job_entrepreneur
Feature Importance
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
balance
age
day
campaign
education

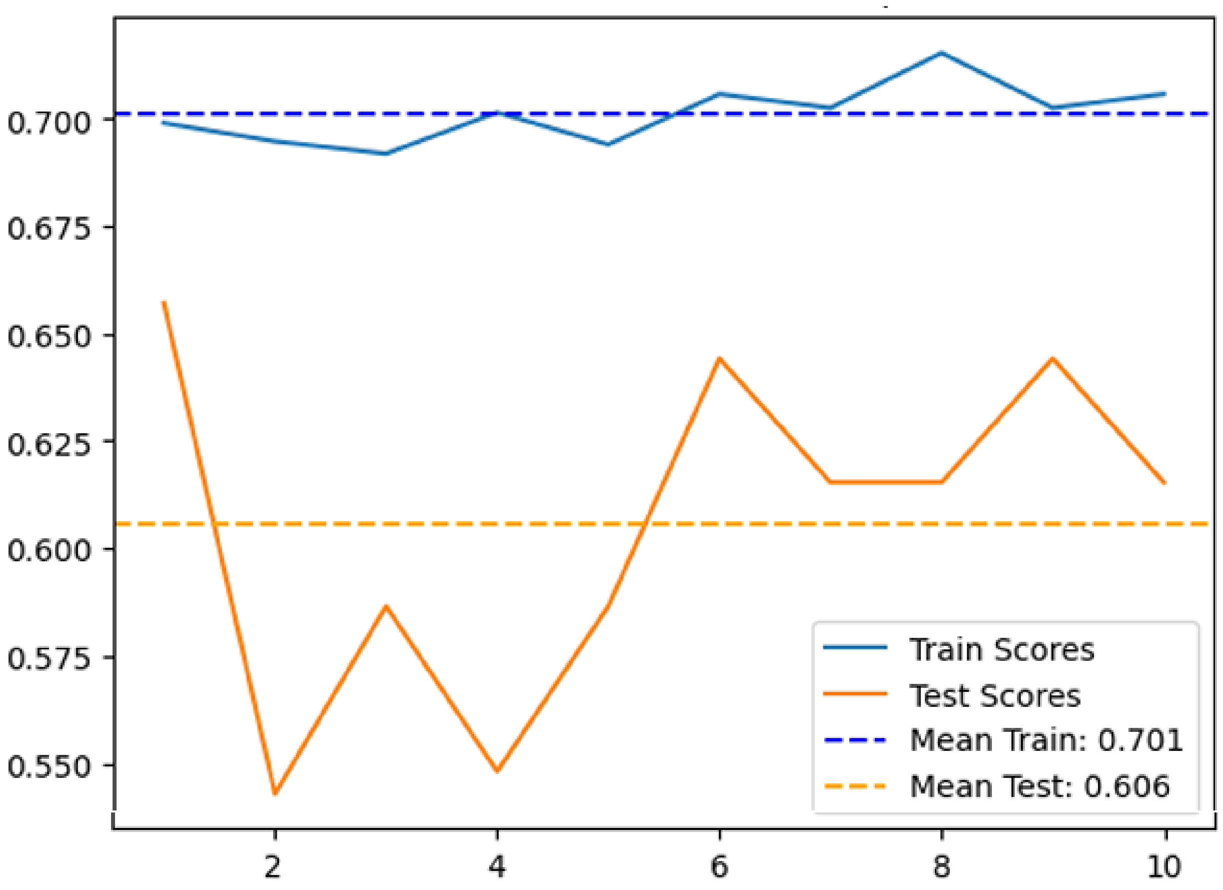
Random Forest Scores with 5 features – Mutual Info

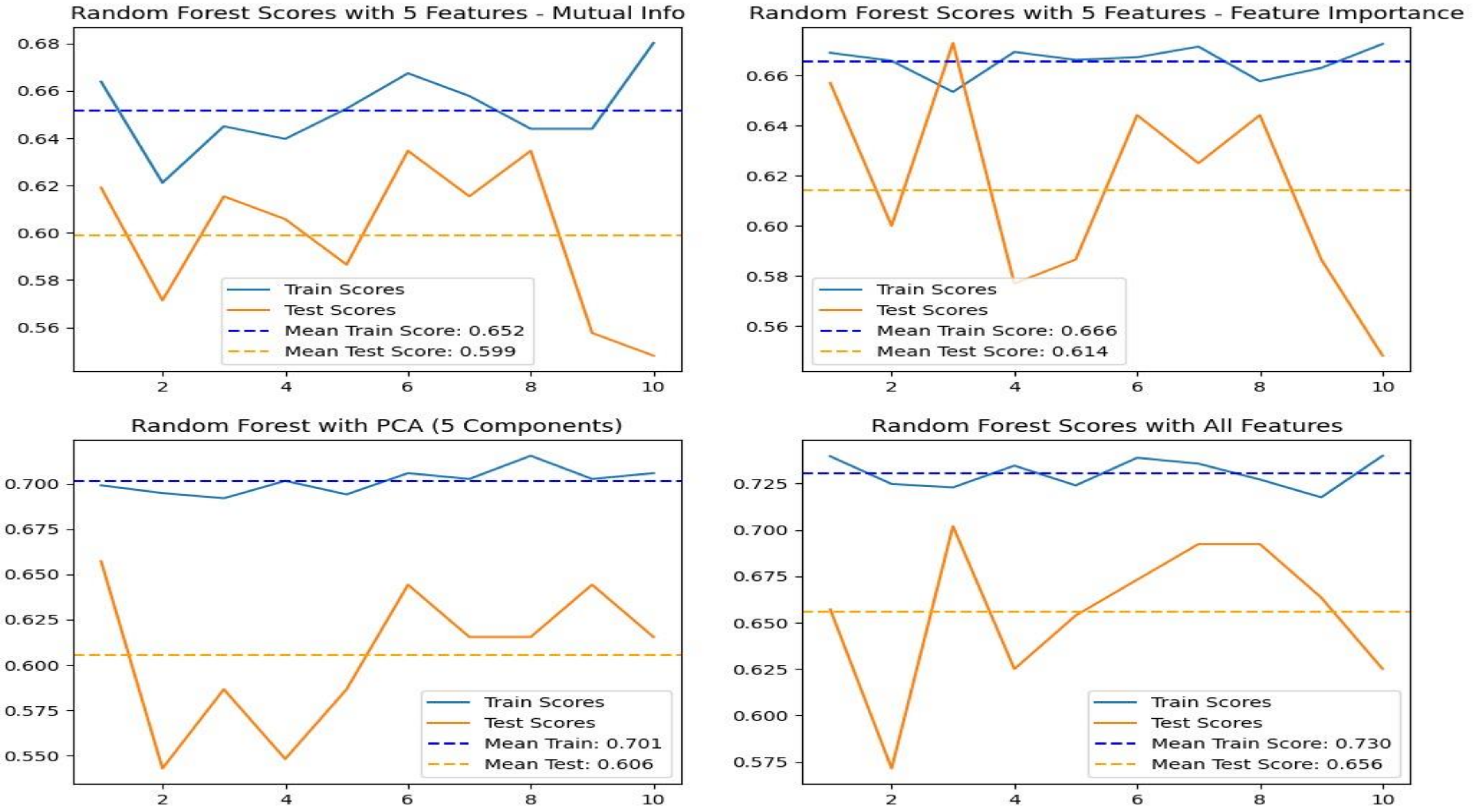


Random Forest Scores with 5 features – feature Importance

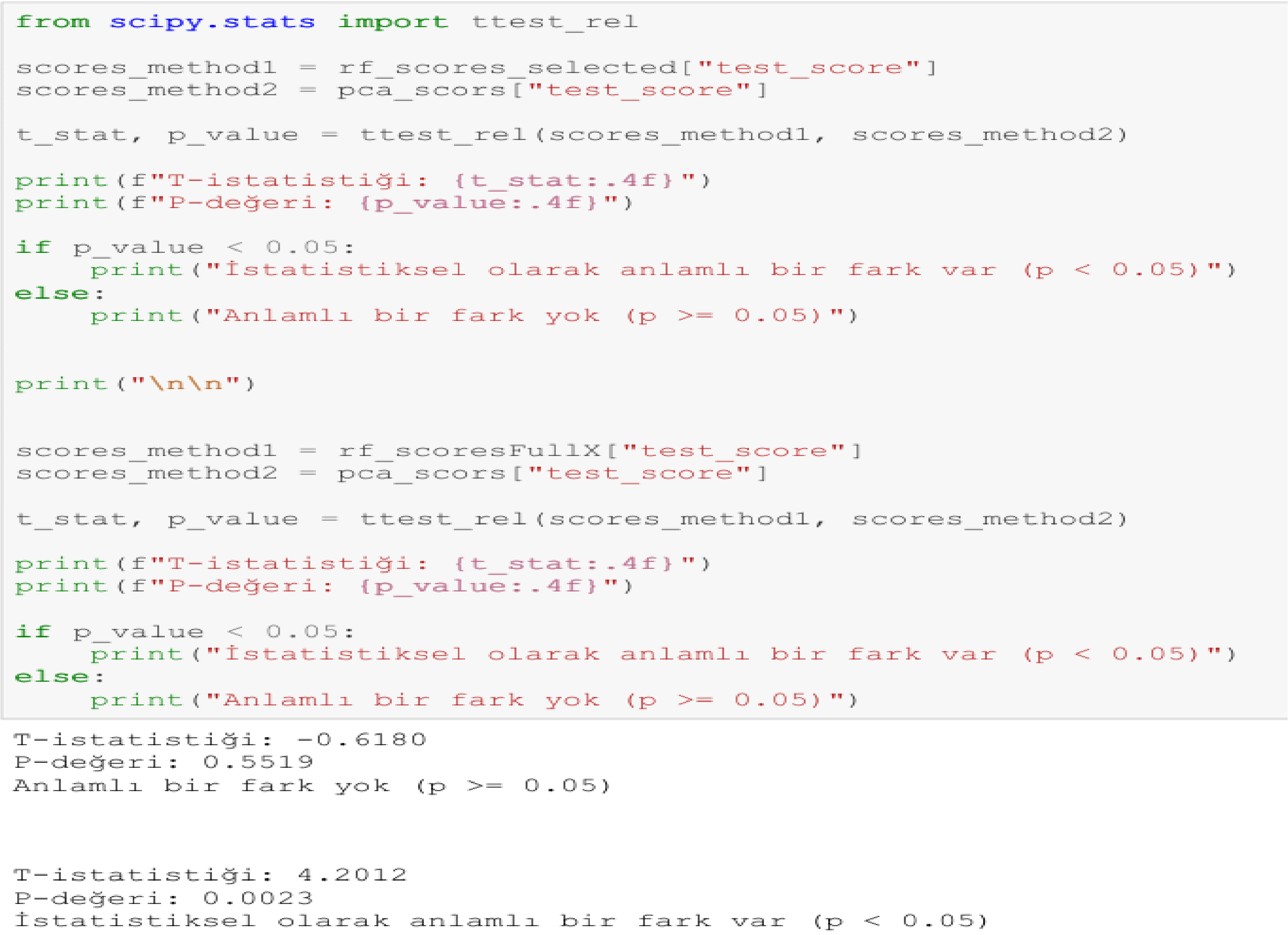


Random Forest With PCA (5 COMPONENTS)





Random Forest Scores with 5 Features - Mutual Info
0.68
0.66
0.64
0.62
0.60
0.58
0.56
2
4
6
8
10
Train Scores
Test Scores
Mean Train Score: 0.652
Mean Test Score: 0.599
Random Forest Scores with 5 Features - Feature Importance
0.66
0.64
0.62
0.60
0.58
0.56
2
4
6
8
10
Train Scores
Test Scores
Mean Train Score: 0.666
Mean Test Score: 0.614
Random Forest with PCA (5 Components)
0.700
0.675
0.650
0.625
0.600
0.575
0.550
2
4
6
8
10
Train Scores
Test Scores
Mean Train: 0.701
Mean Test: 0.606
Random Forest Scores with All Features
0.725
0.700
0.675
0.650
0.625
0.600
0.575
2
4
6
8
10
Train Scores
Test Scores
Mean Train Score: 0.730
Mean Test Score: 0.656



```python
from scipy.stats import ttest_rel

scores_method1 = rf_scores_selected["test_score"]
scores_method2 = pca_scors["test_score"]

t_stat, p_value = ttest_rel(scores_method1, scores_method2)

print(f"T-istatistiği: {t_stat:.4f}")
print(f"P-değeri: {p_value:.4f}")

if p_value < 0.05:
    print("İstatistiksel olarak anlamlı bir fark var (p < 0.05)")
else:
    print("Anlamlı bir fark yok (p >= 0.05)")


print("\n\n")


scores_method1 = rf_scoresFullX["test_score"]
scores_method2 = pca_scors["test_score"]

t_stat, p_value = ttest_rel(scores_method1, scores_method2)

print(f"T-istatistiği: {t_stat:.4f}")
print(f"P-değeri: {p_value:.4f}")

if p_value < 0.05:
    print("İstatistiksel olarak anlamlı bir fark var (p < 0.05)")
else:
    print("Anlamlı bir fark yok (p >= 0.05)")
```

```
T-istatistiği: -0.6180
P-değeri: 0.5519
Anlamlı bir fark yok (p >= 0.05)



T-istatistiği: 4.2012
P-değeri: 0.0023
İstatistiksel olarak anlamlı bir fark var (p < 0.05)
```