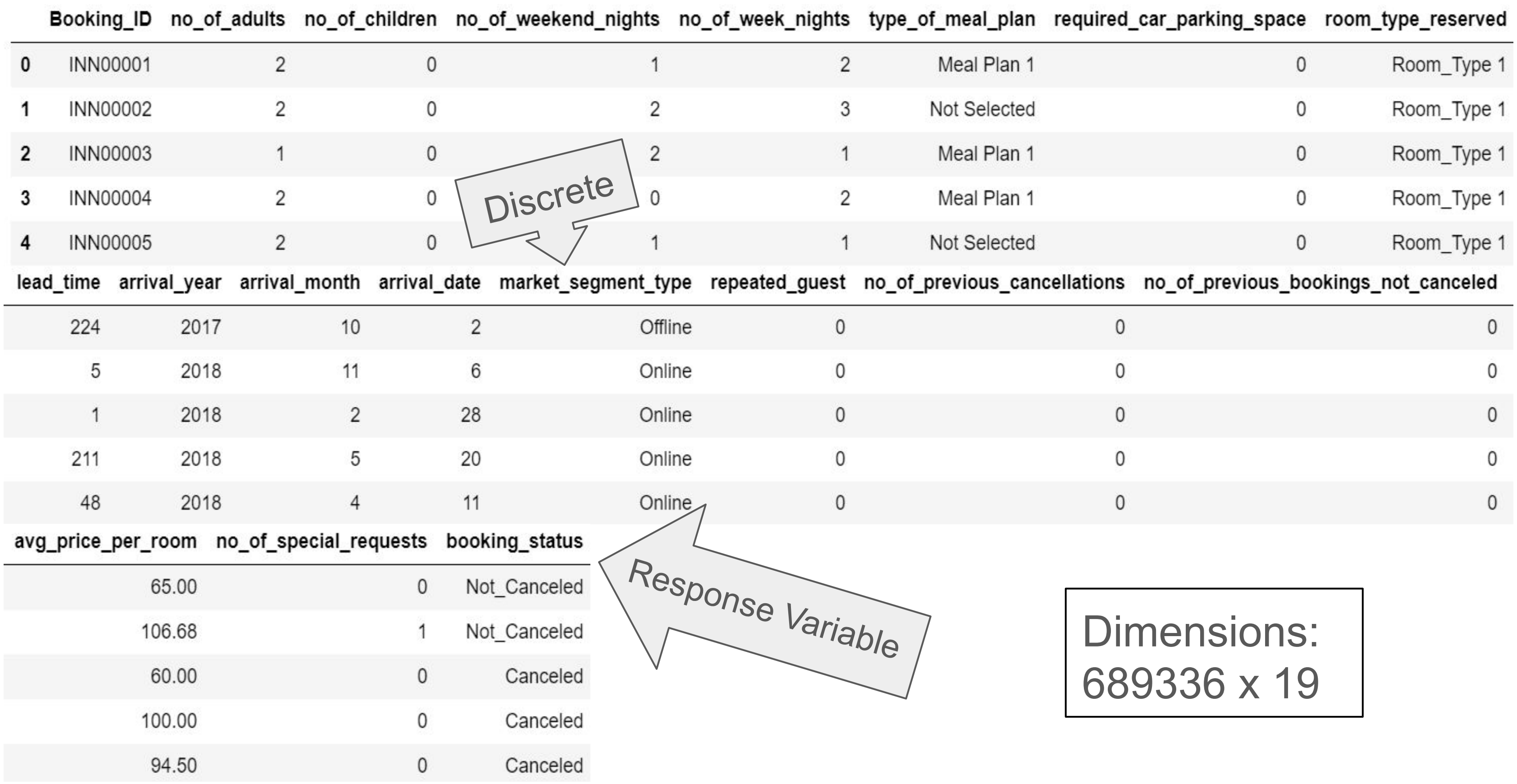
# Hotel ML Model

Eric Fontaine, Jonathan McGechie, Jake Warmflash

# hotel.head()



|   | Booking_ID | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | type_of_meal_plan | required_car_parking_space | room_type_reserved | lead_time | arrival_year | arrival_month | arrival_date | market_segment_type | repeated_guest | no_of_previous_cancellations | no_of_previous_bookings_not_canceled | avg_price_per_room | no_of_special_requests | booking_status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | INN00001 | 2 | 0 | 1 | 2 | Meal Plan 1 | 0 | Room_Type 1 | 224 | 2017 | 10 | 2 | Offline | 0 | 0 | 0 | 65.00 | 0 | Not_Canceled |
| **1** | INN00002 | 2 | 0 | 2 | 3 | Not Selected | 0 | Room_Type 1 | 5 | 2018 | 11 | 6 | Online | 0 | 0 | 0 | 106.68 | 1 | Not_Canceled |
| **2** | INN00003 | 1 | 0 | 2 | 1 | Meal Plan 1 | 0 | Room_Type 1 | 1 | 2018 | 2 | 28 | Online | 0 | 0 | 0 | 60.00 | 0 | Canceled |
| **3** | INN00004 | 2 | 0 | 0 | 2 | Meal Plan 1 | 0 | Room_Type 1 | 211 | 2018 | 5 | 20 | Online | 0 | 0 | 0 | 100.00 | 0 | Canceled |
| **4** | INN00005 | 2 | 0 | 1 | 1 | Not Selected | 0 | Room_Type 1 | 48 | 2018 | 4 | 11 | Online | 0 | 0 | 0 | 94.50 | 0 | Canceled |

Discrete

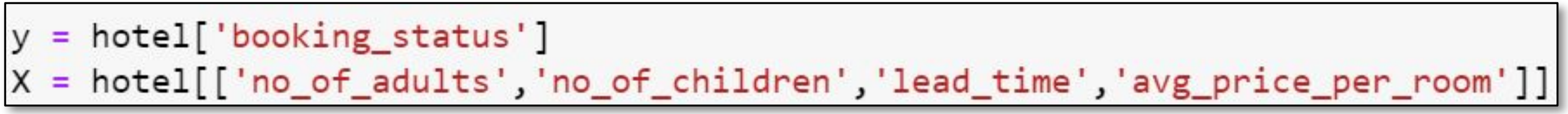
Discrete

Discrete

Response Variable

Dimensions:
689336 x 19

# Random Forest Model



```
y = hotel['booking_status']
X = hotel[['no_of_adults','no_of_children','lead_time','avg_price_per_room']]
```

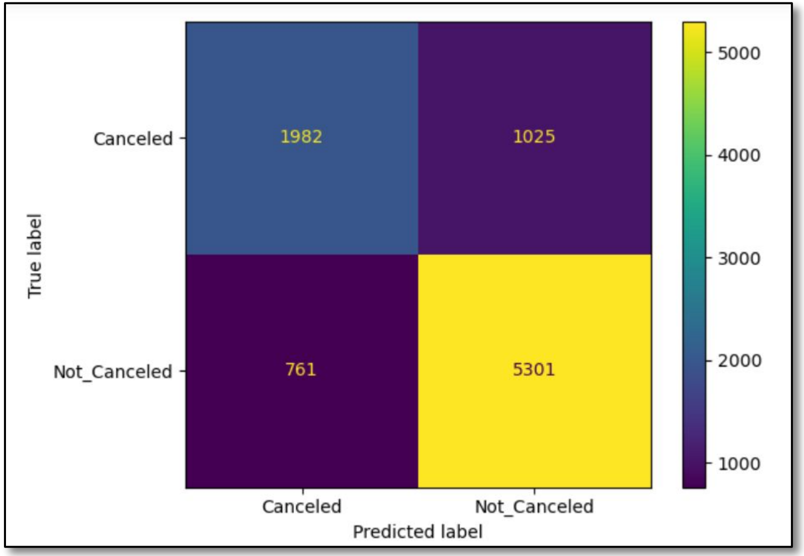
model = RandomForestClassifier(n_estimators=10)

Cohen Kappa Score: 0.545664070375975

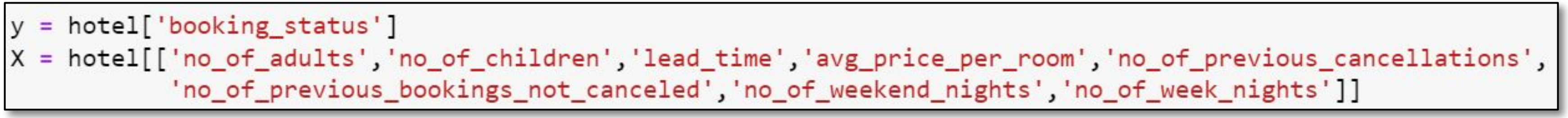
Accuracy: 0.8030653875840776

Precision: 0.72256653

Recall: 0.6591287



# And again…

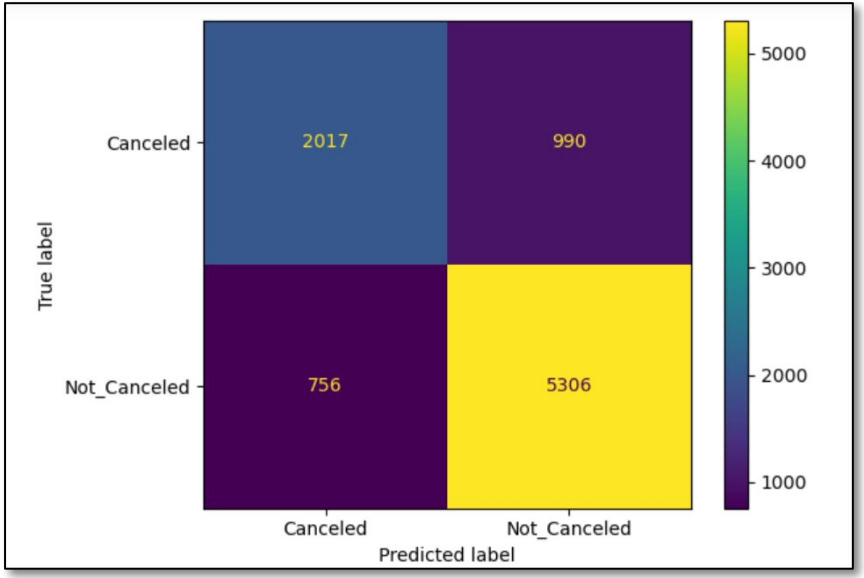


```
y = hotel['booking_status']
X = hotel[['no_of_adults','no_of_children','lead_time','avg_price_per_room','no_of_previous_cancellations',
           'no_of_previous_bookings_not_canceled','no_of_weekend_nights','no_of_week_nights']]
```

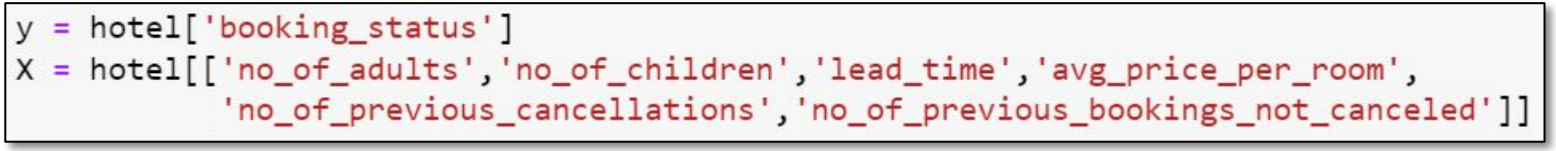
model = RandomForestClassifier(n_estimators=10)

Cohen Kappa Score:
0.5569784878841972

Accuracy:
0.8074760172014555



# Random Forest Model again

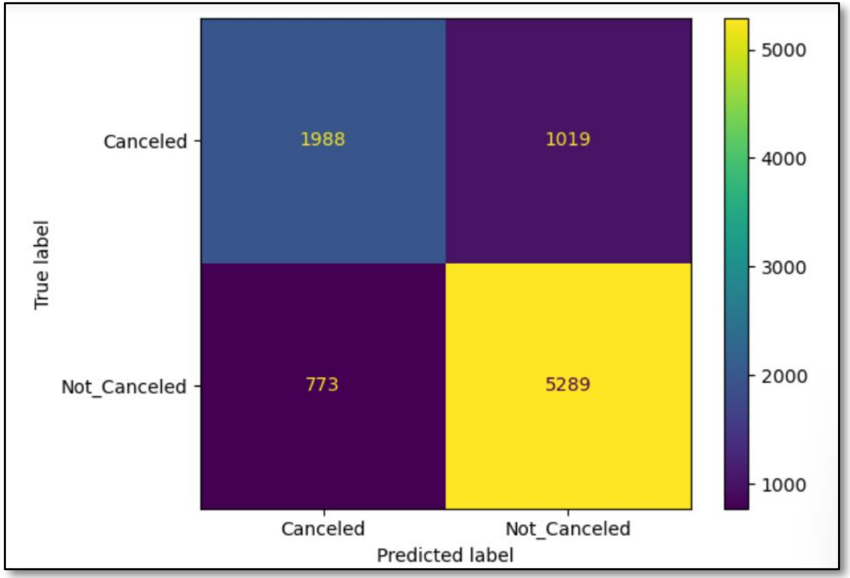


```
y = hotel['booking_status']
X = hotel[['no_of_adults','no_of_children','lead_time','avg_price_per_room',
           'no_of_previous_cancellations','no_of_previous_bookings_not_canceled']]
```

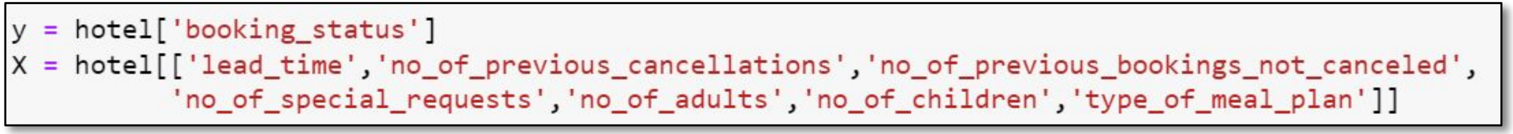
```
model = RandomForestClassifier(n_estimators=10)
```

Cohen Kappa Score:
0.544839821989642

Accuracy:
0.8024037931414709



# This isn’t working

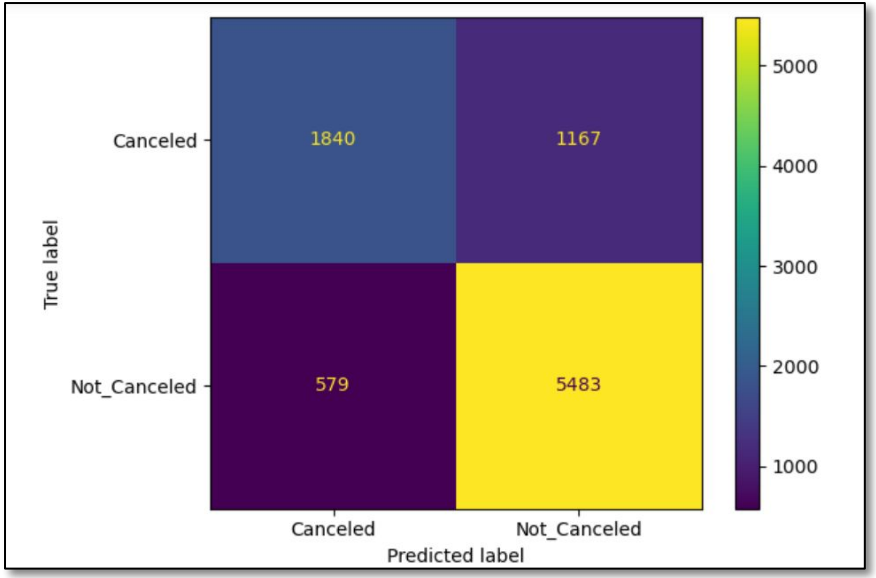


```
y = hotel['booking_status']
X = hotel[['lead_time','no_of_previous_cancellations','no_of_previous_bookings_not_canceled',
          'no_of_special_requests','no_of_adults','no_of_children','type_of_meal_plan']]
```

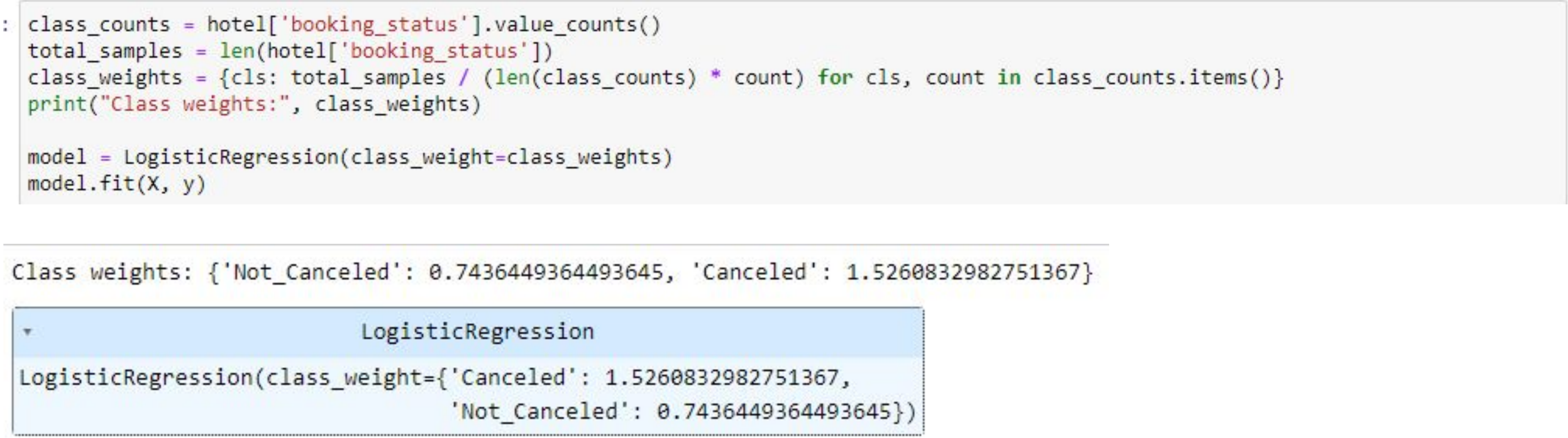
```
model = RandomForestClassifier(n_estimators=10)
```

Cohen Kappa Score:
0.5431554302923487

Accuracy:
0.8074760172014555



# Logistic Regression

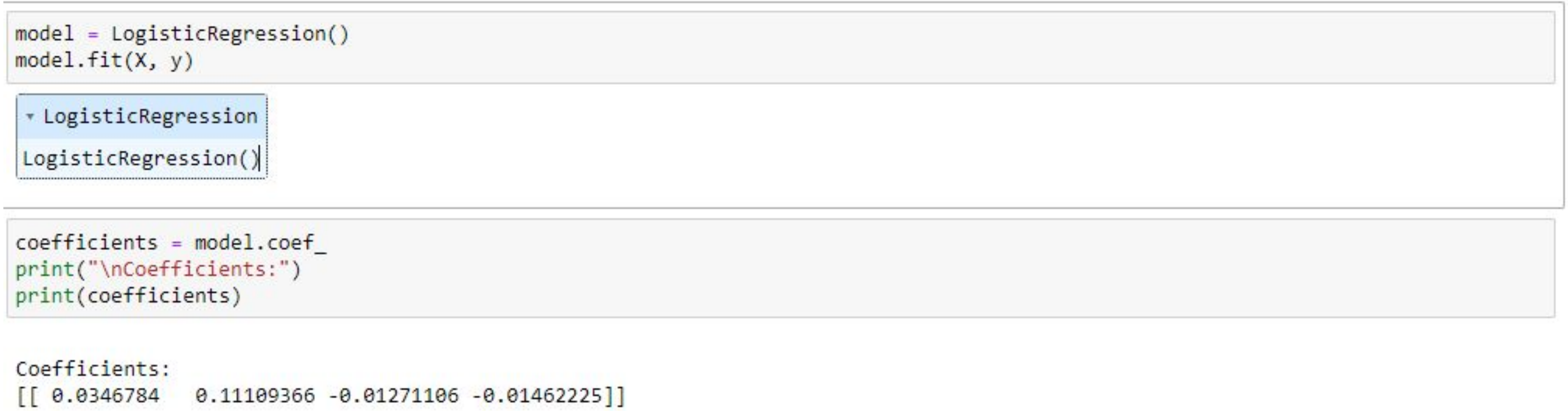


```
class_counts = hotel['booking_status'].value_counts()
total_samples = len(hotel['booking_status'])
class_weights = {cls: total_samples / (len(class_counts) * count) for cls, count in class_counts.items()}
print("Class weights:", class_weights)

model = LogisticRegression(class_weight=class_weights)
model.fit(X, y)
```

```
Class weights: {'Not_Canceled': 0.7436449364493645, 'Canceled': 1.526083298275136 7}
```

```
▾                           LogisticRegression
LogisticRegression(class_weight={'Canceled': 1.526083298275136 7,
                                 'Not_Canceled': 0.7436449364493645})
```

# Fitting model for Logistic Regression



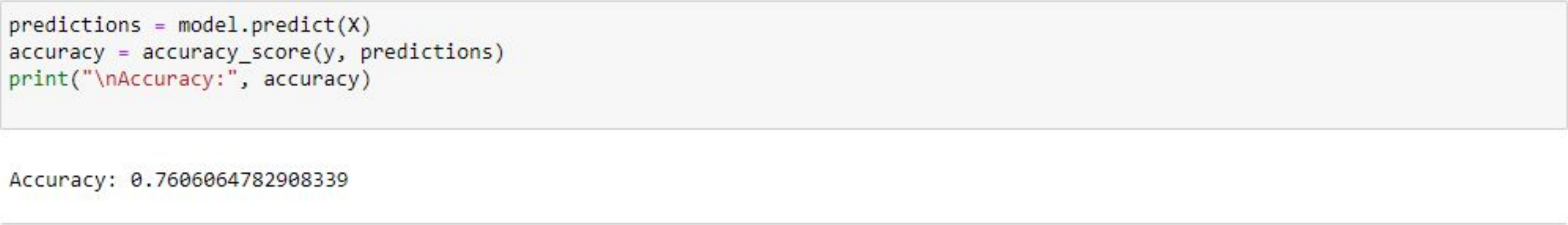
```
model = LogisticRegression()
model.fit(X, y)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
coefficients = model.coef_
print("\nCoefficients:")
print(coefficients)
```

```

Coefficients:
[[ 0.0346784   0.11109366 -0.01271106 -0.01462225]]
```

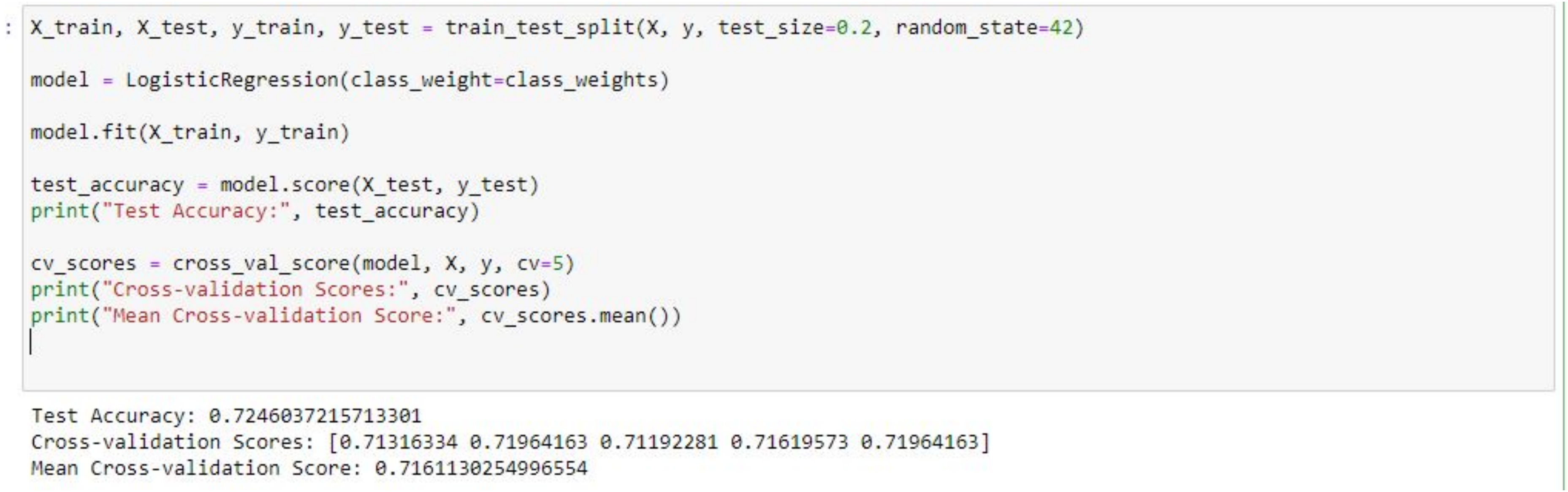
# Predictions Accuracy not the best results.



```python
predictions = model.predict(X)
accuracy = accuracy_score(y, predictions)
print("\nAccuracy:", accuracy)
```

```
Accuracy: 0.7606064782908339
```

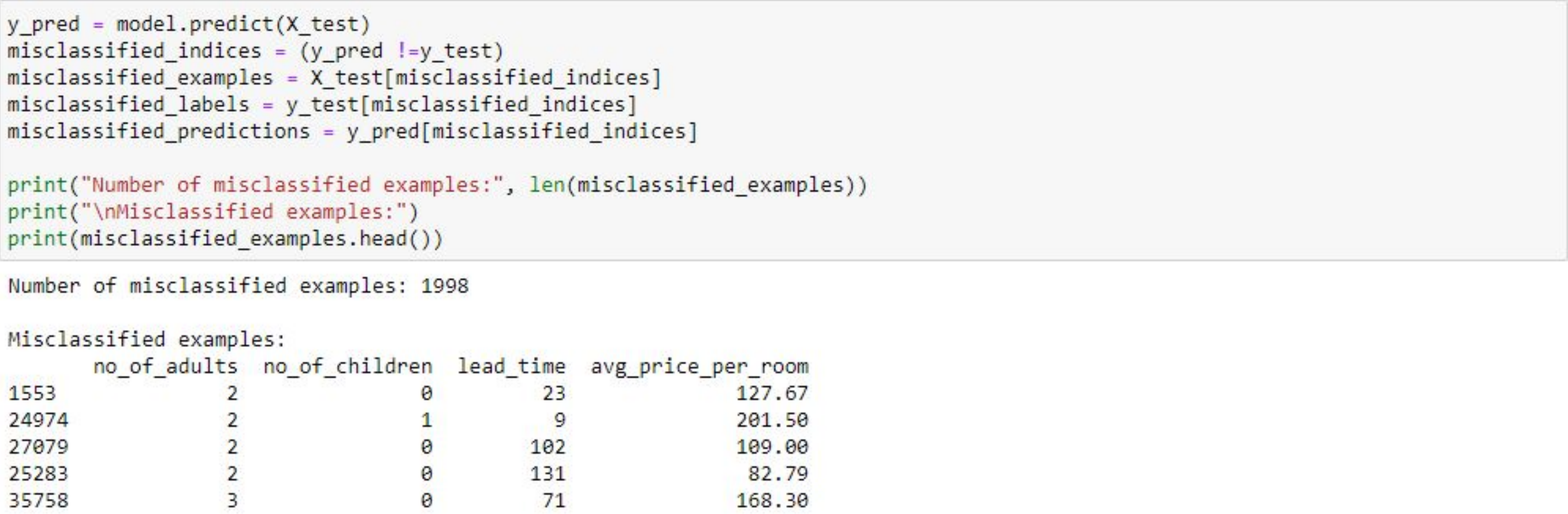
# Running X_train for the Logistic Regression



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LogisticRegression(class_weight=class_weights)

model.fit(X_train, y_train)

test_accuracy = model.score(X_test, y_test)
print("Test Accuracy:", test_accuracy)

cv_scores = cross_val_score(model, X, y, cv=5)
print("Cross-validation Scores:", cv_scores)
print("Mean Cross-validation Score:", cv_scores.mean())
```

```
Test Accuracy: 0.7246037215713301
Cross-validation Scores: [0.71316334 0.71964163 0.71192281 0.71619573 0.71964163]
Mean Cross-validation Score: 0.7161130254996554
```

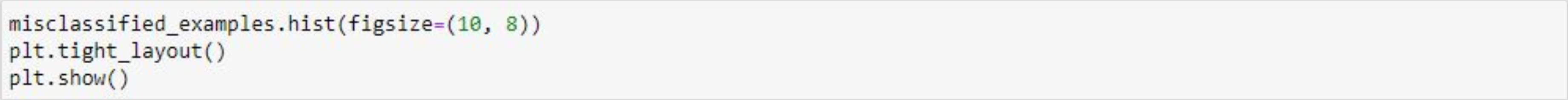
# Wondering why the accrucary went down. Misclassification?



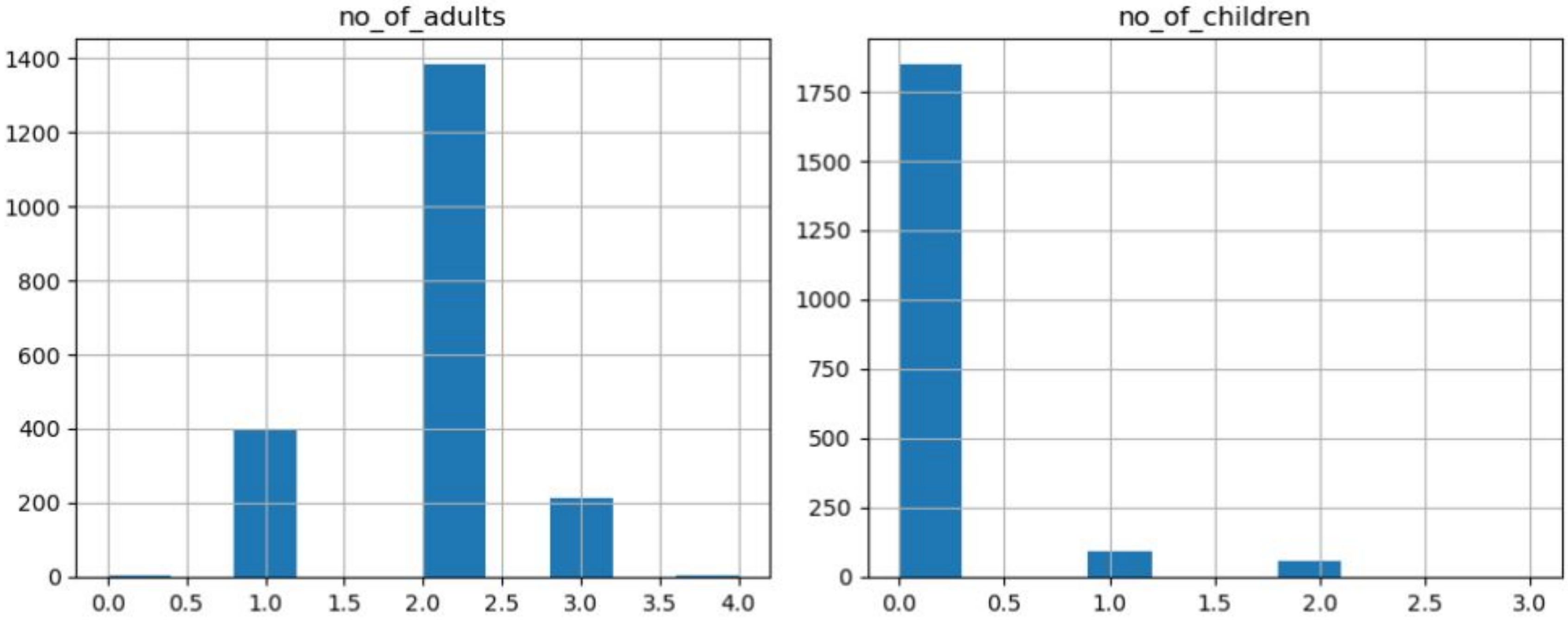
```
y_pred = model.predict(X_test)
misclassified_indices = (y_pred !=y_test)
misclassified_examples = X_test[misclassified_indices]
misclassified_labels = y_test[misclassified_indices]
misclassified_predictions = y_pred[misclassified_indices]

print("Number of misclassified examples:", len(misclassified_examples))
print("\nMisclassified examples:")
print(misclassified_examples.head())
```

```
Number of misclassified examples: 1998

Misclassified examples:
       no_of_adults  no_of_children  lead_time  avg_price_per_room
1553              2               0         23              127.67
24974             2               1          9              201.50
27079             2               0        102              109.00
25283             2               0        131               82.79
35758             3               0         71              168.30
```

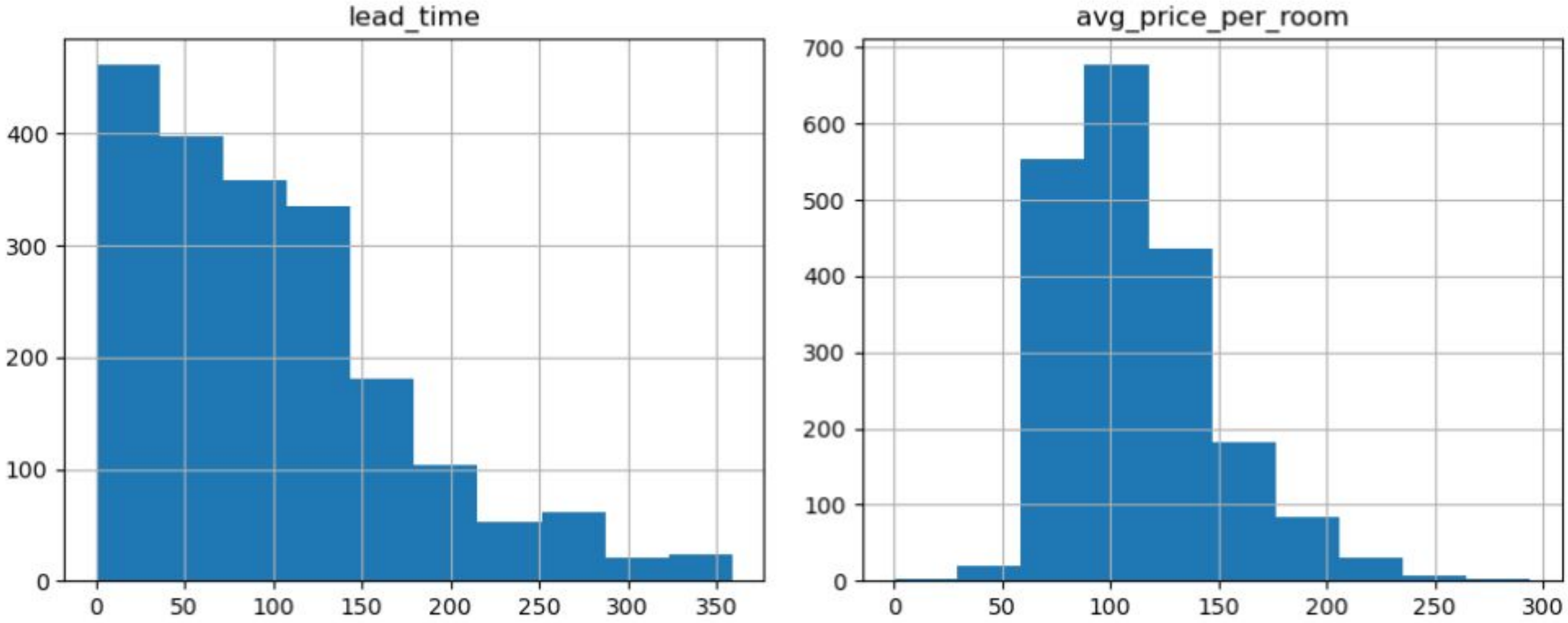
# Misclassification?  Graphing the misclassified_examples



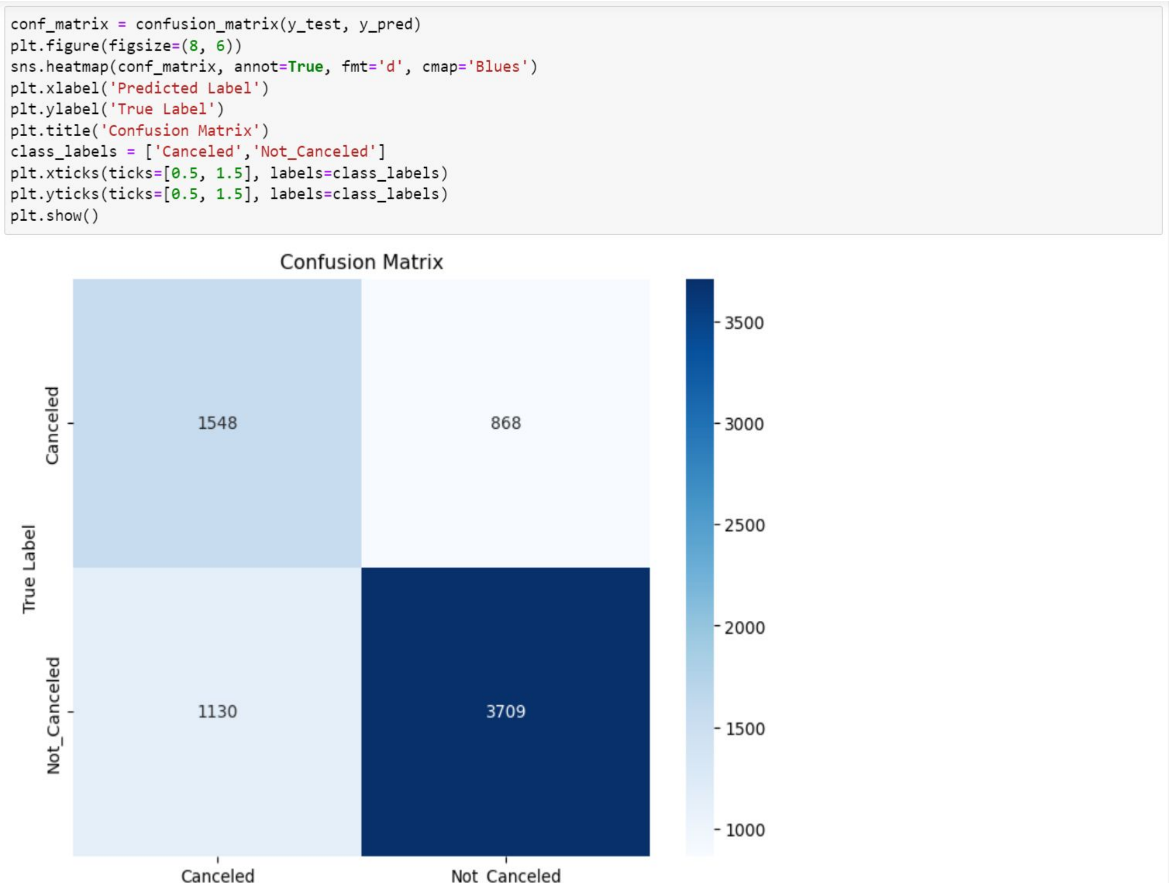
```
misclassified_examples.hist(figsize=(10, 8))
plt.tight_layout()
plt.show()
```



# Misclassification? Graphing the misclassified_examples continued



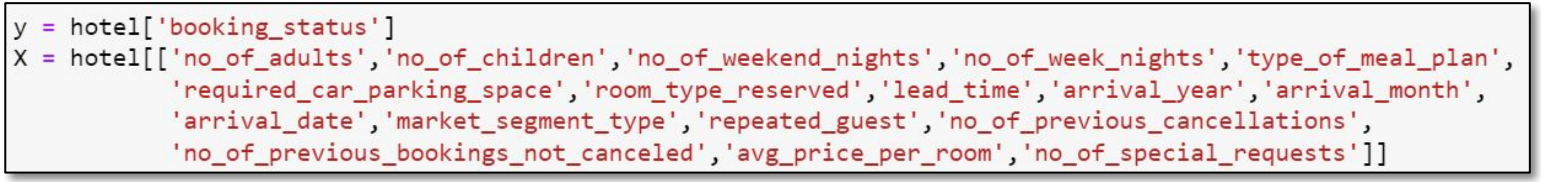
# What does the Confusion Matrix show for Logistic Regression



```
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
class_labels = ['Canceled','Not_Canceled']
plt.xticks(ticks=[0.5, 1.5], labels=class_labels)
plt.yticks(ticks=[0.5, 1.5], labels=class_labels)
plt.show()
```

Precision:
0.5780433
Recall:
0.6407284

# Lets try a Feature Selector with a Transformer



```
y = hotel['booking_status']
X = hotel[['no_of_adults','no_of_children','no_of_weekend_nights','no_of_week_nights','type_of_meal_plan',
           'required_car_parking_space','room_type_reserved','lead_time','arrival_year','arrival_month',
           'arrival_date','market_segment_type','repeated_guest','no_of_previous_cancellations',
           'no_of_previous_bookings_not_canceled','avg_price_per_room','no_of_special_requests']]
```

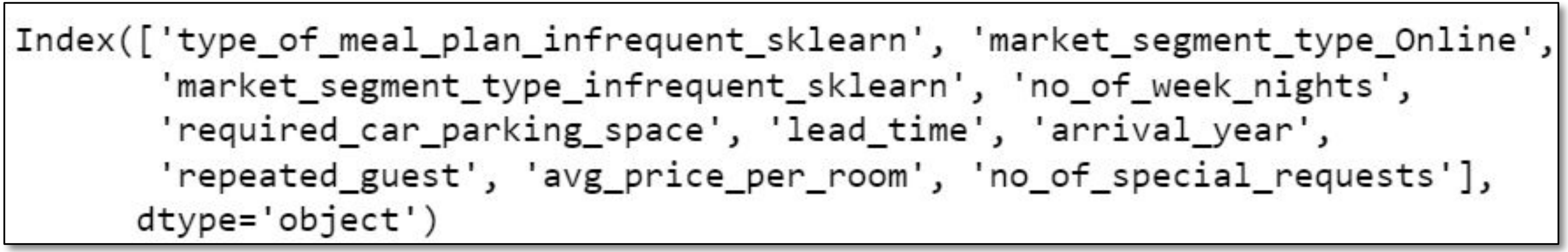
```
transformer = ColumnTransformer(
        [('categories', encoder, ['type_of_meal_plan', 'room_type_reserved', 'market_segment_type']),
        ('scaled_price_points', StandardScaler(),
['no_of_adults','no_of_children','no_of_weekend_nights','no_of_week_nights',
                                        'required_car_parking_space','lead_time','arrival_year','arrival_month',
                                        'arrival_date','repeated_guest','no_of_previous_cancellations',
                                        'no_of_previous_bookings_not_canceled','avg_price_per_room',
                                        'no_of_special_requests'])],
        remainder='drop', verbose_feature_names_out=False)
```

```
transformer.fit(X_train, y_train)
```

# Lets try a Feature Selector with a Transformer

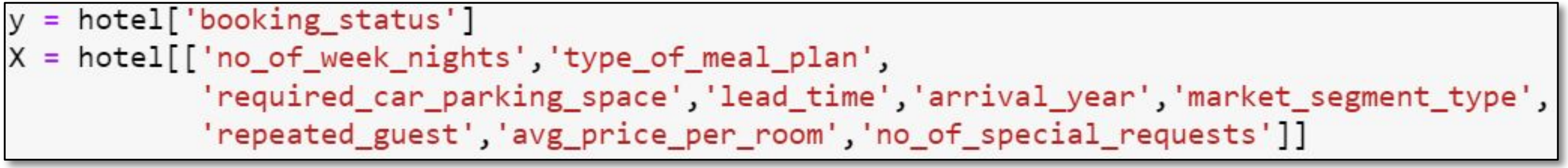
```
feature_selector = SelectKBest(k=10)
```

```
X_train_trans_df.columns[feature_selector.get_support()]
```



```
Index(['type_of_meal_plan_infrequent_sklearn', 'market_segment_type_Online',
       'market_segment_type_infrequent_sklearn', 'no_of_week_nights',
       'required_car_parking_space', 'lead_time', 'arrival_year',
       'repeated_guest', 'avg_price_per_room', 'no_of_special_requests'],
      dtype='object')
```

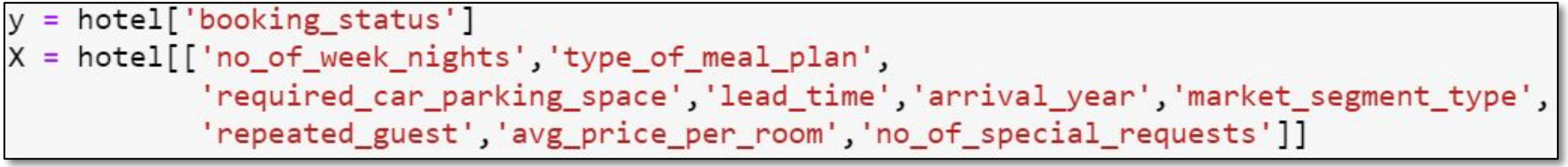
# Using Those Chosen Features…



```
y = hotel['booking_status']
X = hotel[['no_of_week_nights','type_of_meal_plan',
           'required_car_parking_space','lead_time','arrival_year','market_segment_type',
           'repeated_guest','avg_price_per_room','no_of_special_requests']]
```

```
transformer = ColumnTransformer(
        [('categories', encoder, ['type_of_meal_plan', 'market_segment_type']),
        ('scaled_hotel', StandardScaler(), ['no_of_week_nights',
           'required_car_parking_space','lead_time','arrival_year',
           'repeated_guest','avg_price_per_room','no_of_special_requests'])],
        remainder='drop', verbose_feature_names_out=False)
```

# Add a Pipeline…



```
y = hotel['booking_status']
X = hotel[['no_of_week_nights','type_of_meal_plan',
          'required_car_parking_space','lead_time','arrival_year','market_segment_type',
          'repeated_guest','avg_price_per_room','no_of_special_requests']]
```

```
classification_pipeline = Pipeline([('hotel_transformer', transformer),
                                    ('RF_model', RandomForestClassifier())])
```

```
classification_pipeline.fit(X_train, y_train)
y_pred = classification_pipeline.predict(X_test)
```

```
Accuracy = 0.8797000771860183
```

Much Better!

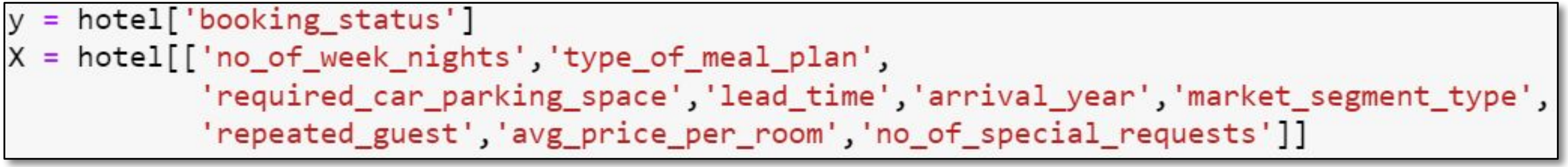
# Hyperparameters?

```
y = hotel['booking_status']
X = hotel[['no_of_week_nights','type_of_meal_plan',
           'required_car_parking_space','lead_time','arrival_year','market_segment_type',
           'repeated_guest','avg_price_per_room','no_of_special_requests']]
```

```
parameters = {
    'hotel_transformer__categories__max_categories': randint(3, 30),
    'RF_model__max_depth': randint(3, 30),
    'RF_model__min_samples_leaf': randint(2, 10),
    'RF_model__n_estimators': randint(50, 200),
    'RF_model__min_samples_split': randint(2, 10)}
```

```
n_iter_search = 100
random_search = RandomizedSearchCV(classification_pipeline,
param_distributions=parameters, n_iter=n_iter_search, n_jobs=-1)
```

# This gives us



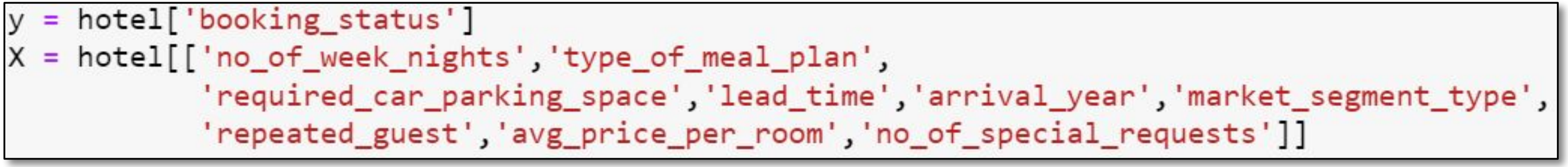
```
y = hotel['booking_status']
X = hotel[['no_of_week_nights','type_of_meal_plan',
           'required_car_parking_space','lead_time','arrival_year','market_segment_type',
           'repeated_guest','avg_price_per_room','no_of_special_requests']]
```

random_search.best_score_ = 0.8821259234755762

random_search.best_estimator_.get_params()

Chosen Hyperparameters:
max_categories=25
max_depth=25
min_samples_leaf=2
min_samples_split=3
n_estimators=143

# Hyperparameters and Selected Features

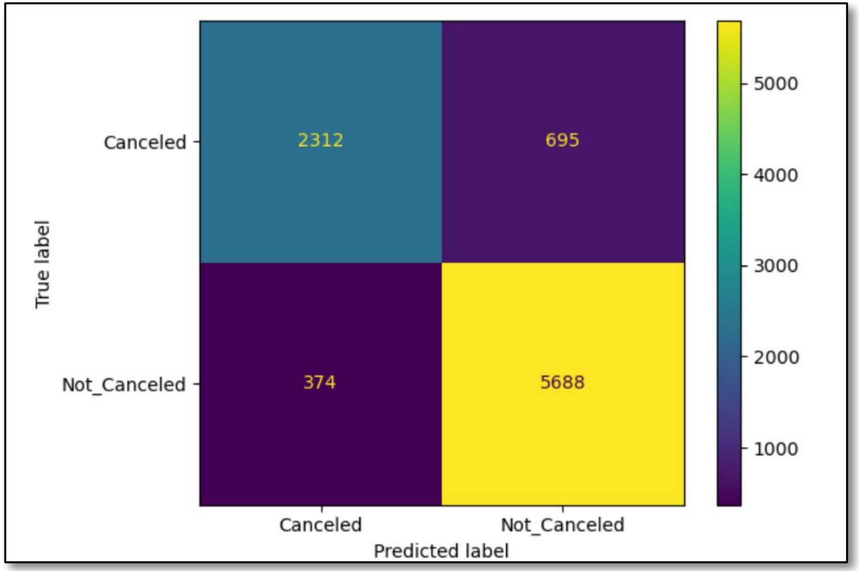


```
y = hotel['booking_status']
X = hotel[['no_of_week_nights','type_of_meal_plan',
           'required_car_parking_space','lead_time','arrival_year','market_segment_type',
           'repeated_guest','avg_price_per_room','no_of_special_requests']]
```

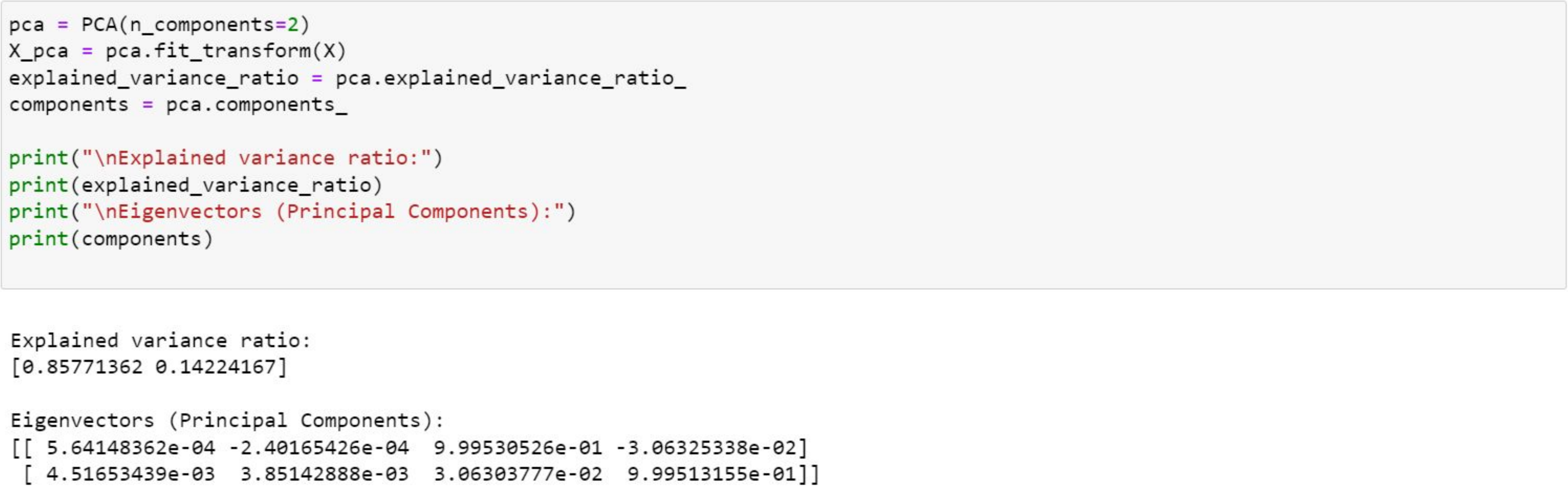
y_pred=random_search.predict(X_test)
Accuracy: 0.8821259234755762

Chosen Hyperparameters:
max_categories=25
max_depth=25
min_samples_leaf=2
min_samples_split=3
n_estimators=143

Precision:
0.8607595
Recall:
0.7688726



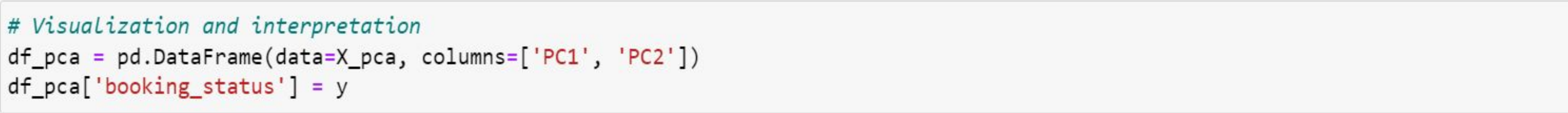
# PCA with Eigenvectors (Linear Regression)



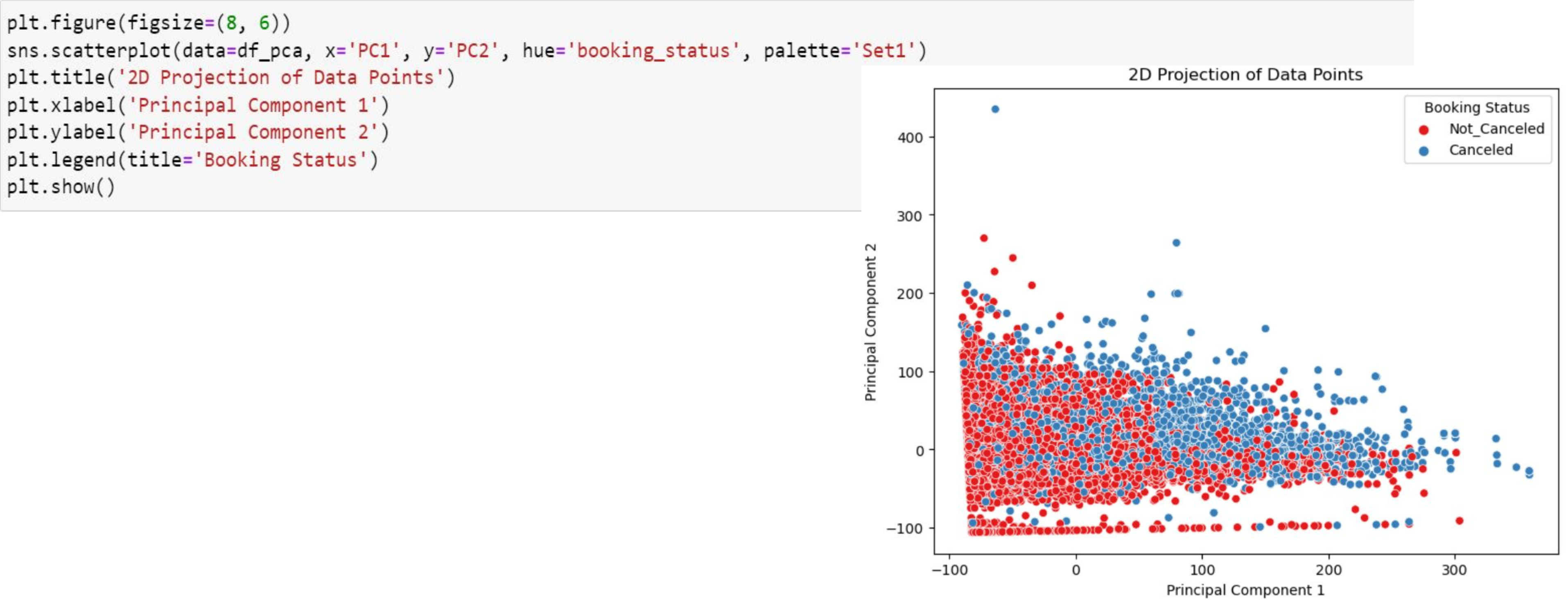
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
explained_variance_ratio = pca.explained_variance_ratio_
components = pca.components_

print("\nExplained variance ratio:")
print(explained_variance_ratio)
print("\nEigenvectors (Principal Components):")
print(components)
```

```
Explained variance ratio:
[0.85771362 0.14224167]

Eigenvectors (Principal Components):
[[ 5.64148362e-04 -2.40165426e-04  9.99530526e-01 -3.06325338e-02]
 [ 4.51653439e-03  3.85142888e-03  3.06303777e-02  9.99513155e-01]]
```

# PCA and 2D Visualization

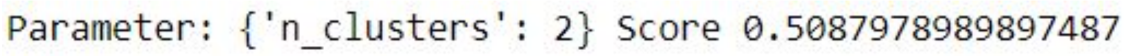


```
# Visualization and interpretation
df_pca = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
df_pca['booking_status'] = y
```



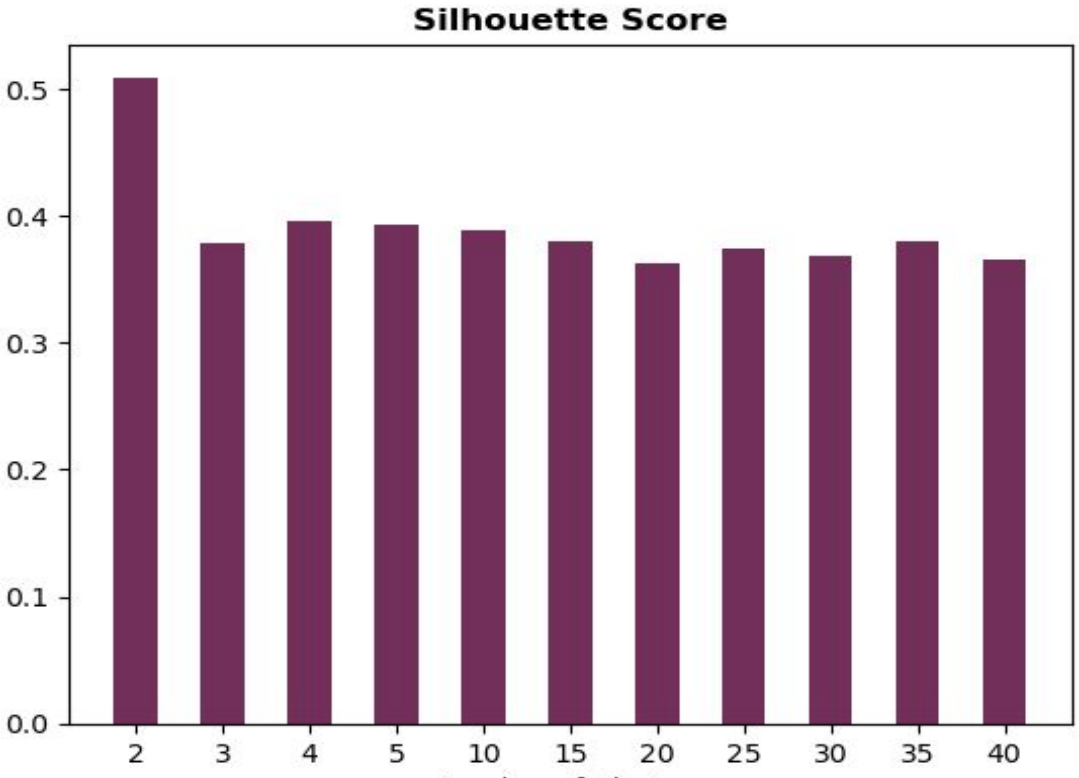
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df_pca, x='PC1', y='PC2', hue='booking_status', palette='Set1')
plt.title('2D Projection of Data Points')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Booking Status')
plt.show()
```

# K mean Clustering & Silhouette Scores

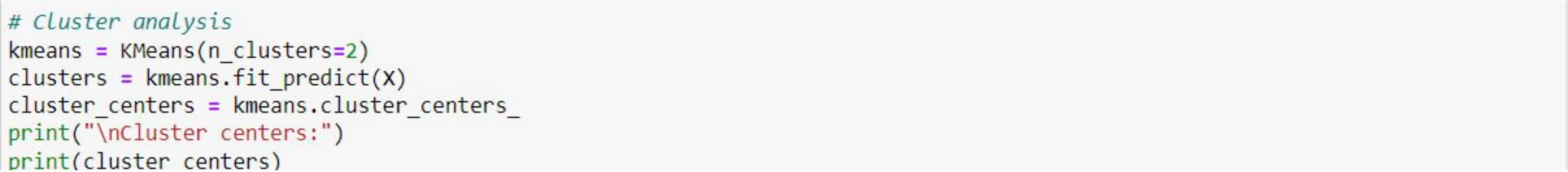


```
Parameter: {'n_clusters': 2} Score 0.5087978989897487
```

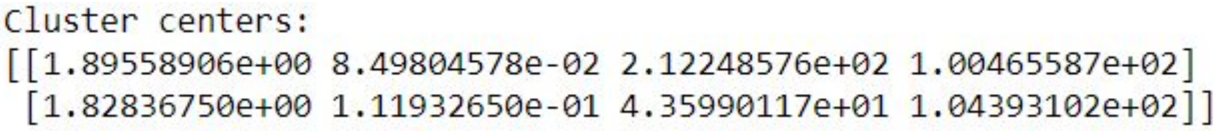
```
from sklearn import metrics
from sklearn.model_selection import
ParameterGrid
```



# Cluster Analysis (KMeans)

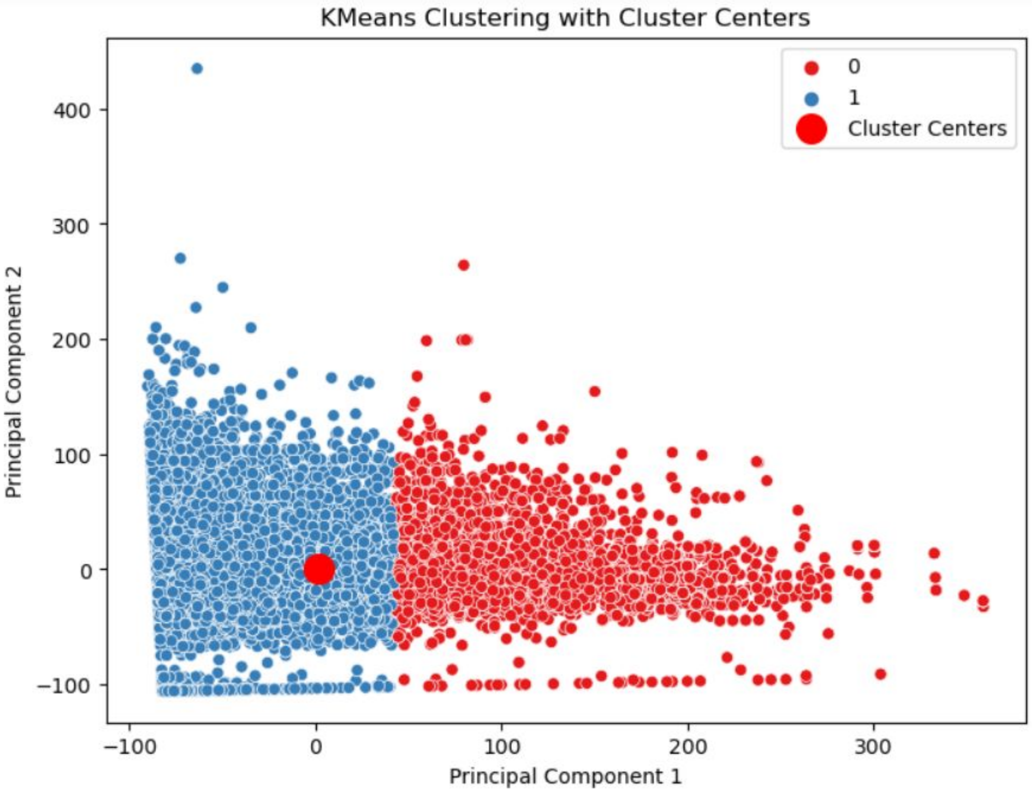


```
# Cluster analysis
kmeans = KMeans(n_clusters=2)
clusters = kmeans.fit_predict(X)
cluster_centers = kmeans.cluster_centers_
print("\nCluster centers:")
print(cluster_centers)
```



```
Cluster centers:
[[1.89558906e+00 8.49804578e-02 2.12248576e+02 1.00465587e+02]
 [1.82836750e+00 1.11932650e-01 4.35990117e+01 1.04393102e+02]]
```

# Cluster Analysis (KMeans) Cluster Graph


KMeans Clustering with Cluster Centers
Cluster Centers
Principal Component 2
Principal Component 1