HOTEL BOOKING PREDICTION

Meliane Ayebah
Matricola: A04402

SOMMARIO

INTRODUZIONE

EXPLORATION DATA ANALYSIS (EDA)

DATA PRE-PROCESSING

MODEL BUILDING

MODELS COMPARISON

CONCLUSIONE

INTRODUZIONE

L’argomento trattato per il progetto finale è hotel booking prediction.

L’obiettivo è quello di creare degli stimatori significativi a partire dal dataset a disposizione e selezionare il modello migliore per predire la prenotazione confrontandolo con i punteggi di accuratezza di diversi modelli di Machine Learning.

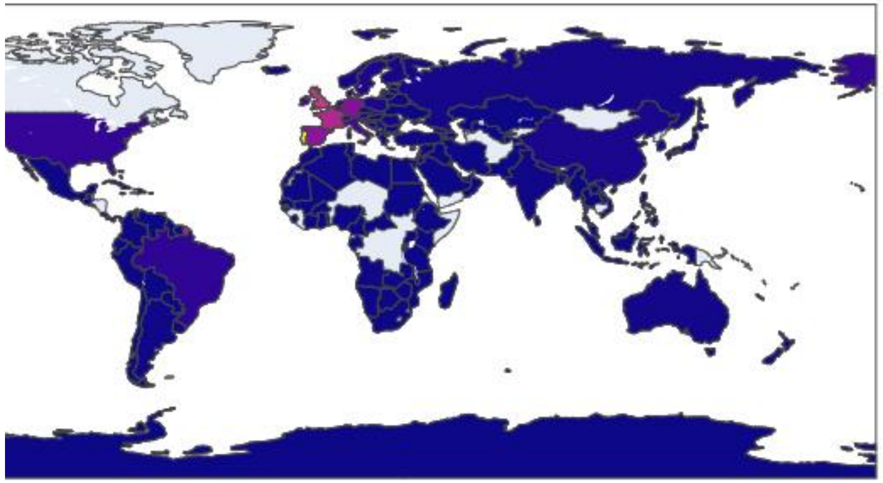
EXPLORATION DATA ANALYSIS (EDA)

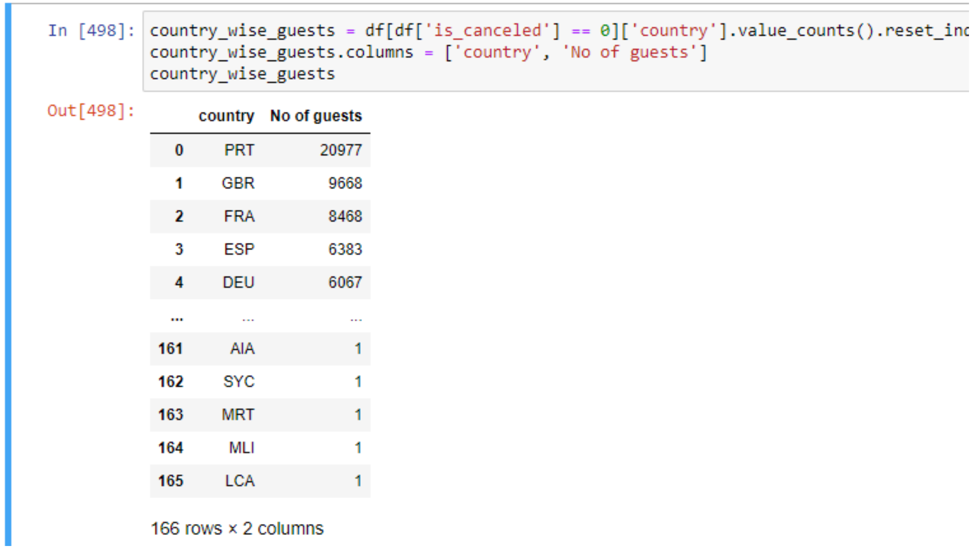
Nella fase di exploration si cercherà a rispondere ai seguenti quesiti:

- Da dove proviene la maggior parte degli ospiti?
- Quanto pagano gli ospiti a notte per una camera ?
- Come varia il prezzo a notte di una camera durante l’anno?

DA DOVE PROVIENE LA MAGGIOR PARTE DEGLI OSPITI?

La maggior parte degli ospiti soggiornano nei due hotel (Hotel City e Resort Hotel) e sono tutti provenienti per la maggior parte dal Portogallo e dagli altri paesi europei.





```
In [498]: country_wise_guests = df[df['is_canceled'] == 0]['country'].value_counts().reset_ind
          country_wise_guests.columns = ['country', 'No of guests']
          country_wise_guests
```

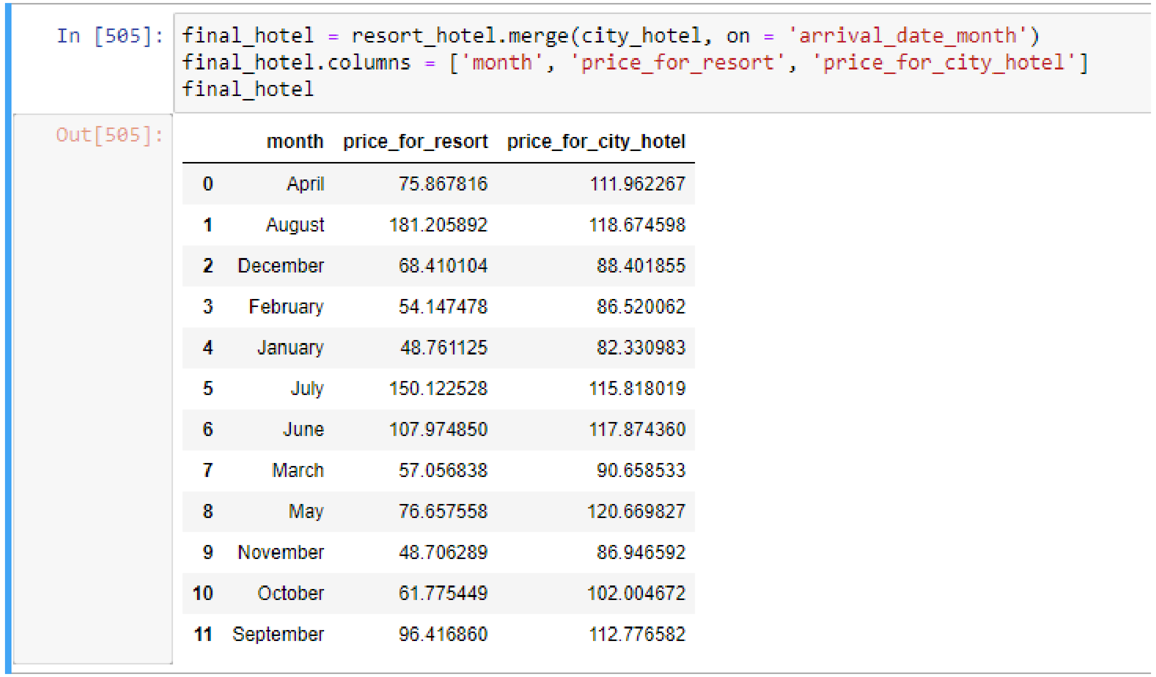
Out[498]:

	country	No of guests
0	PRT	20977
1	GBR	9668
2	FRA	8468
3	ESP	6383
4	DEU	6067
...	...	...
161	AIA	1
162	SYC	1
163	MRT	1
164	MLI	1
165	LCA	1

166 rows × 2 columns

COME VARIA IL PREZZO A NOTTE DI UNA CAMERA DURANTE L’ANNO E QUANTO GLI OSPITI PAGANO UNA CAMERA A NOTTE?

Questa tabella mostra chiaramente che i prezzi del Resort Hotel sono molto più alti durante l’estate mentre i prezzi dell’Hotel city variano di meno e sono più alti durante la primavera e l'autunno.



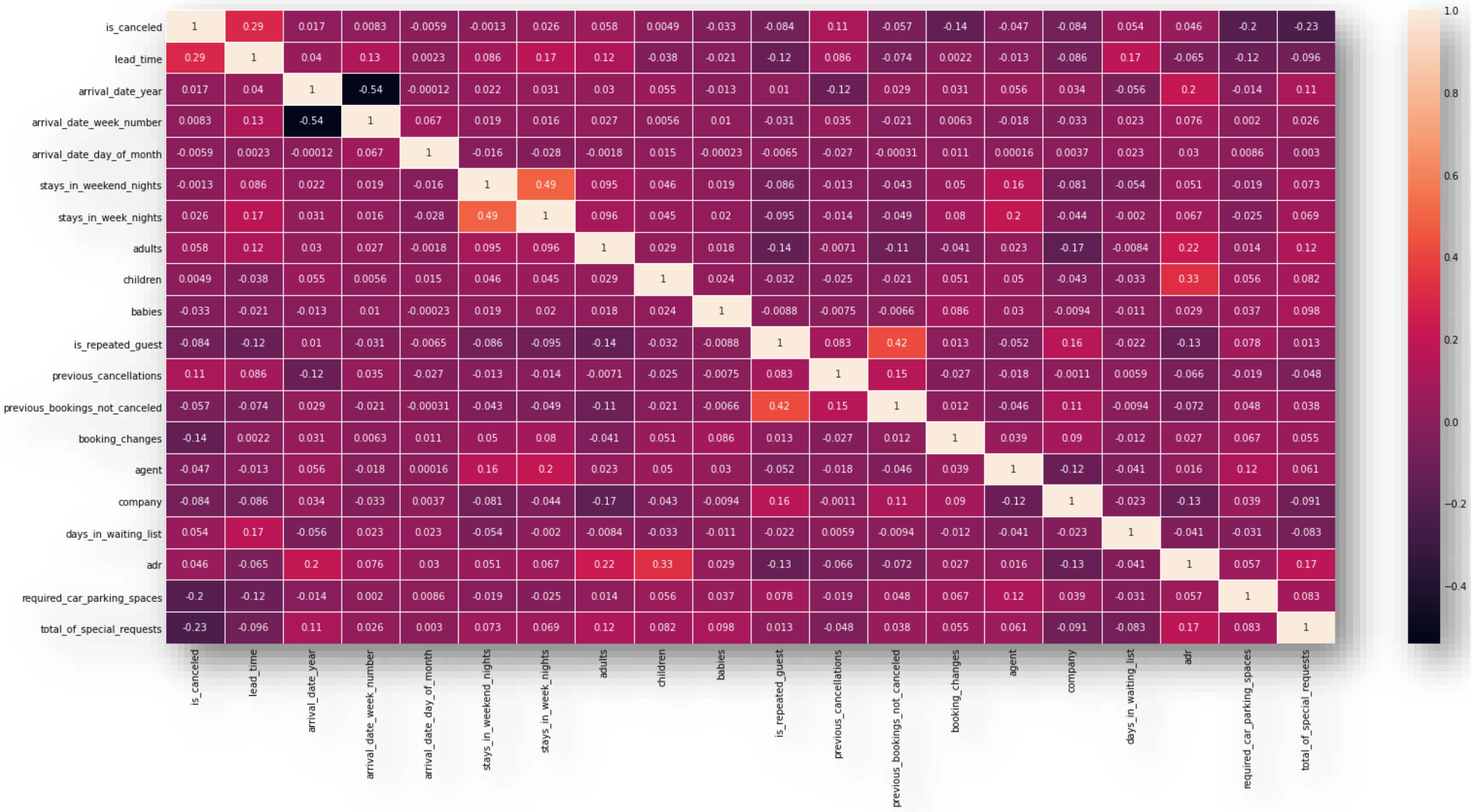
In [505]:
```
final_hotel = resort_hotel.merge(city_hotel, on = 'arrival_date_month')
final_hotel.columns = ['month', 'price_for_resort', 'price_for_city_hotel']
final_hotel
```

Out[505]:

	month	price_for_resort	price_for_city_hotel
0	April	75.867816	111.962267
1	August	181.205892	118.674598
2	December	68.410104	88.401855
3	February	54.147478	86.520062
4	January	48.761125	82.330983
5	July	150.122528	115.818019
6	June	107.974850	117.874360
7	March	57.056838	90.658533
8	May	76.657558	120.669827
9	November	48.706289	86.946592
10	October	61.775449	102.004672
11	September	96.416860	112.776582

DATA PRE-PROCESSING

In questa slide si osserva che le features con alta correlazione sono tutte collocate in corrispondenza della diagonale della piramide.



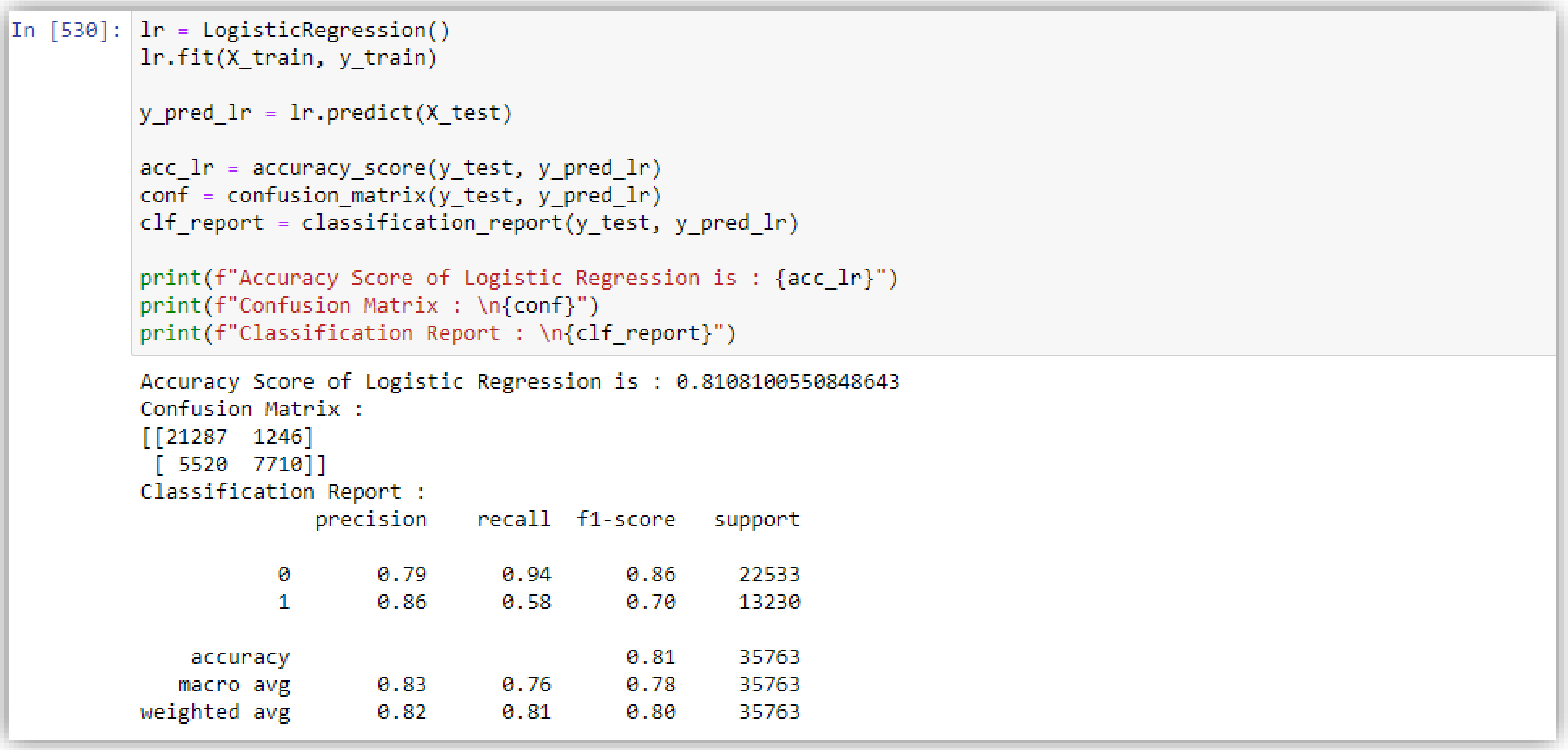
	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	children	babies	is_repeated_guest	previous_cancellations	previous_bookings_not_canceled	booking_changes	agent	company	days_in_waiting_list	adr	required_car_parking_spaces	total_of_special_requests
is_canceled	1	0.29	0.017	0.0083	-0.0059	-0.0013	0.026	0.058	0.0049	-0.033	-0.084	0.11	-0.057	-0.14	-0.047	-0.084	0.054	0.046	-0.2	-0.23
lead_time	0.29	1	0.04	0.13	0.0023	0.086	0.17	0.12	-0.038	-0.021	-0.12	0.086	-0.074	0.0022	-0.013	-0.086	0.17	-0.065	-0.12	-0.096
arrival_date_year	0.017	0.04	1	-0.54	-0.00012	0.022	0.031	0.03	0.055	-0.013	0.01	-0.12	0.029	0.031	0.056	0.034	-0.056	0.2	-0.014	0.11
arrival_date_week_number	0.0083	0.13	-0.54	1	0.067	0.019	0.016	0.027	0.0056	0.01	-0.031	0.035	-0.021	0.0063	-0.018	-0.033	0.023	0.076	0.002	0.026
arrival_date_day_of_month	-0.0059	0.0023	-0.00012	0.067	1	-0.016	-0.028	-0.0018	0.015	-0.00023	-0.0065	-0.027	-0.00031	0.011	0.00016	0.0037	0.023	0.03	0.0086	0.003
stays_in_weekend_nights	-0.0013	0.086	0.022	0.019	-0.016	1	0.49	0.095	0.046	0.019	-0.086	-0.013	-0.043	0.05	0.16	-0.081	-0.054	0.051	-0.019	0.073
stays_in_week_nights	0.026	0.17	0.031	0.016	-0.028	0.49	1	0.096	0.045	0.02	-0.095	-0.014	-0.049	0.08	0.2	-0.044	-0.002	0.067	-0.025	0.069
adults	0.058	0.12	0.03	0.027	-0.0018	0.095	0.096	1	0.029	0.018	-0.14	-0.0071	-0.11	-0.041	0.023	-0.17	-0.0084	0.22	0.014	0.12
children	0.0049	-0.038	0.055	0.0056	0.015	0.046	0.045	0.029	1	0.024	-0.032	-0.025	-0.021	0.051	0.05	-0.043	-0.033	0.33	0.056	0.082
babies	-0.033	-0.021	-0.013	0.01	-0.00023	0.019	0.02	0.018	0.024	1	-0.0088	-0.0075	-0.0066	0.086	0.03	-0.0094	-0.011	0.029	0.037	0.098
is_repeated_guest	-0.084	-0.12	0.01	-0.031	-0.0065	-0.086	-0.095	-0.14	-0.032	-0.0088	1	0.083	0.42	0.013	-0.052	0.16	-0.022	-0.13	0.078	0.013
previous_cancellations	0.11	0.086	-0.12	0.035	-0.027	-0.013	-0.014	-0.0071	-0.025	-0.0075	0.083	1	0.15	-0.027	-0.018	-0.0011	0.0059	-0.066	-0.019	-0.048
previous_bookings_not_canceled	-0.057	-0.074	0.029	-0.021	-0.00031	-0.043	-0.049	-0.11	-0.021	-0.0066	0.42	0.15	1	0.012	-0.046	0.11	-0.0094	-0.072	0.048	0.038
booking_changes	-0.14	0.0022	0.031	0.0063	0.011	0.05	0.08	-0.041	0.051	0.086	0.013	-0.027	0.012	1	0.039	0.09	-0.012	0.027	0.067	0.055
agent	-0.047	-0.013	0.056	-0.018	0.00016	0.16	0.2	0.023	0.05	0.03	-0.052	-0.018	-0.046	0.039	1	-0.12	-0.041	0.016	0.12	0.061
company	-0.084	-0.086	0.034	-0.033	0.0037	-0.081	-0.044	-0.17	-0.043	-0.0094	0.16	-0.0011	0.11	0.09	-0.12	1	-0.023	-0.13	0.039	-0.091
days_in_waiting_list	0.054	0.17	-0.056	0.023	0.023	-0.054	-0.002	-0.0084	-0.033	-0.011	-0.022	0.0059	-0.0094	-0.012	-0.041	-0.023	1	-0.041	-0.031	-0.083
adr	0.046	-0.065	0.2	0.076	0.03	0.051	0.067	0.22	0.33	0.029	-0.13	-0.066	-0.072	0.027	0.016	-0.13	-0.041	1	0.057	0.17
required_car_parking_spaces	-0.2	-0.12	-0.014	0.002	0.0086	-0.019	-0.025	0.014	0.056	0.037	0.078	-0.019	0.048	0.067	0.12	0.039	-0.031	0.057	1	0.083
total_of_special_requests	-0.23	-0.096	0.11	0.026	0.003	0.073	0.069	0.12	0.082	0.098	0.013	-0.048	0.038	0.055	0.061	-0.091	-0.083	0.17	0.083	1

MODELS BUILDING

Per poter valutare il modello migliore e scegliere quello efficiente si andrà ad implementare i seguenti modelli:

- Logistic Regression
- Decision Tree Classifier
- Random Forest
- KNN
- Ada Boost Classifier
- Gradient Boosting Classifier

LOGISTIC REGRESSION

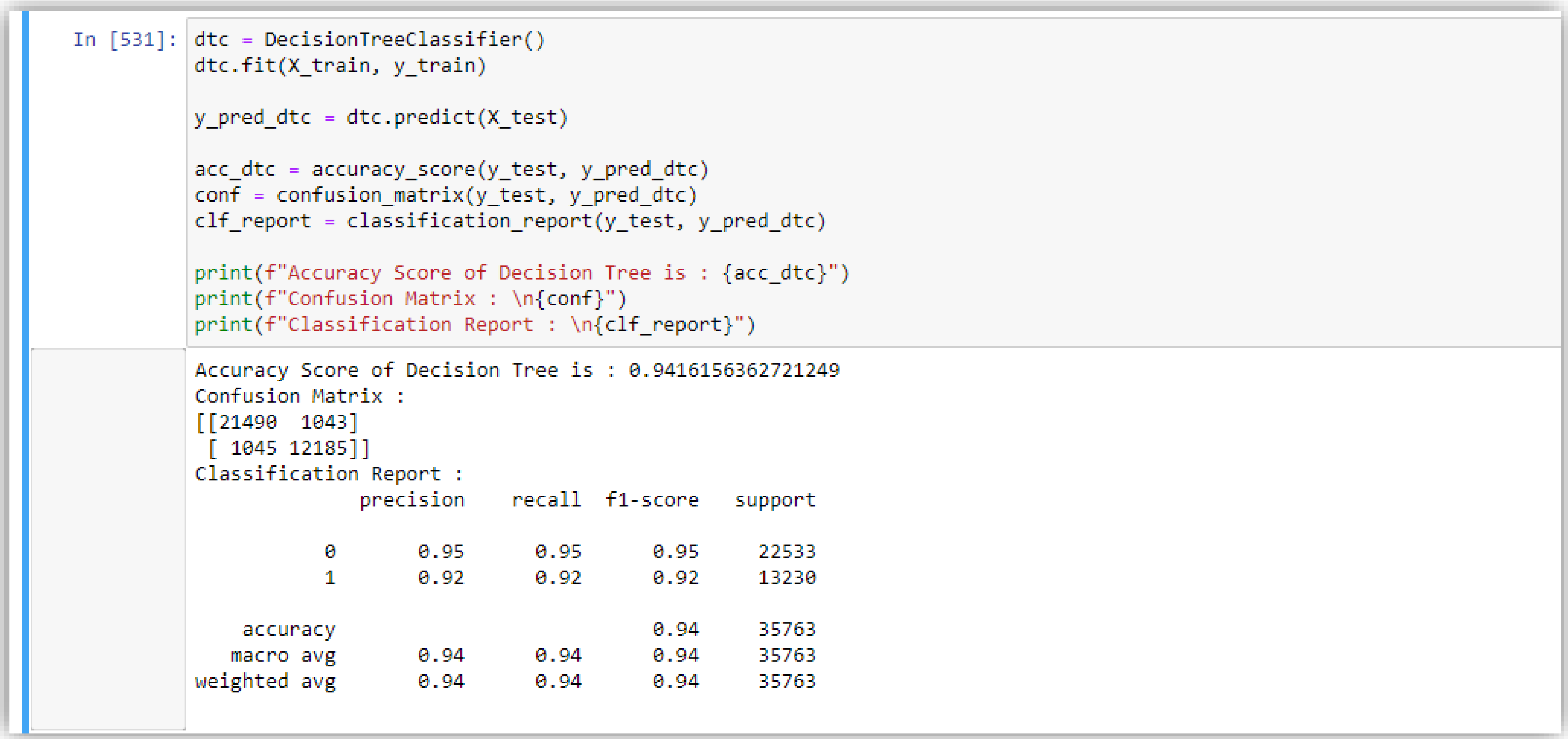


In [530]:

```
lr = LogisticRegression()
lr.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)

acc_lr = accuracy_score(y_test, y_pred_lr)
conf = confusion_matrix(y_test, y_pred_lr)
clf_report = classification_report(y_test, y_pred_lr)

print(f"Accuracy Score of Logistic Regression is : {acc_lr}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Logistic Regression is : 0.8108100550848643
Confusion Matrix : 
[[21287  1246]
 [ 5520  7710]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.79      0.94      0.86     22533
           1       0.86      0.58      0.70     13230

    accuracy                           0.81     35763
   macro avg       0.83      0.76      0.78     35763
weighted avg       0.82      0.81      0.80     35763
```

DECISION TREE CLASSIFIER

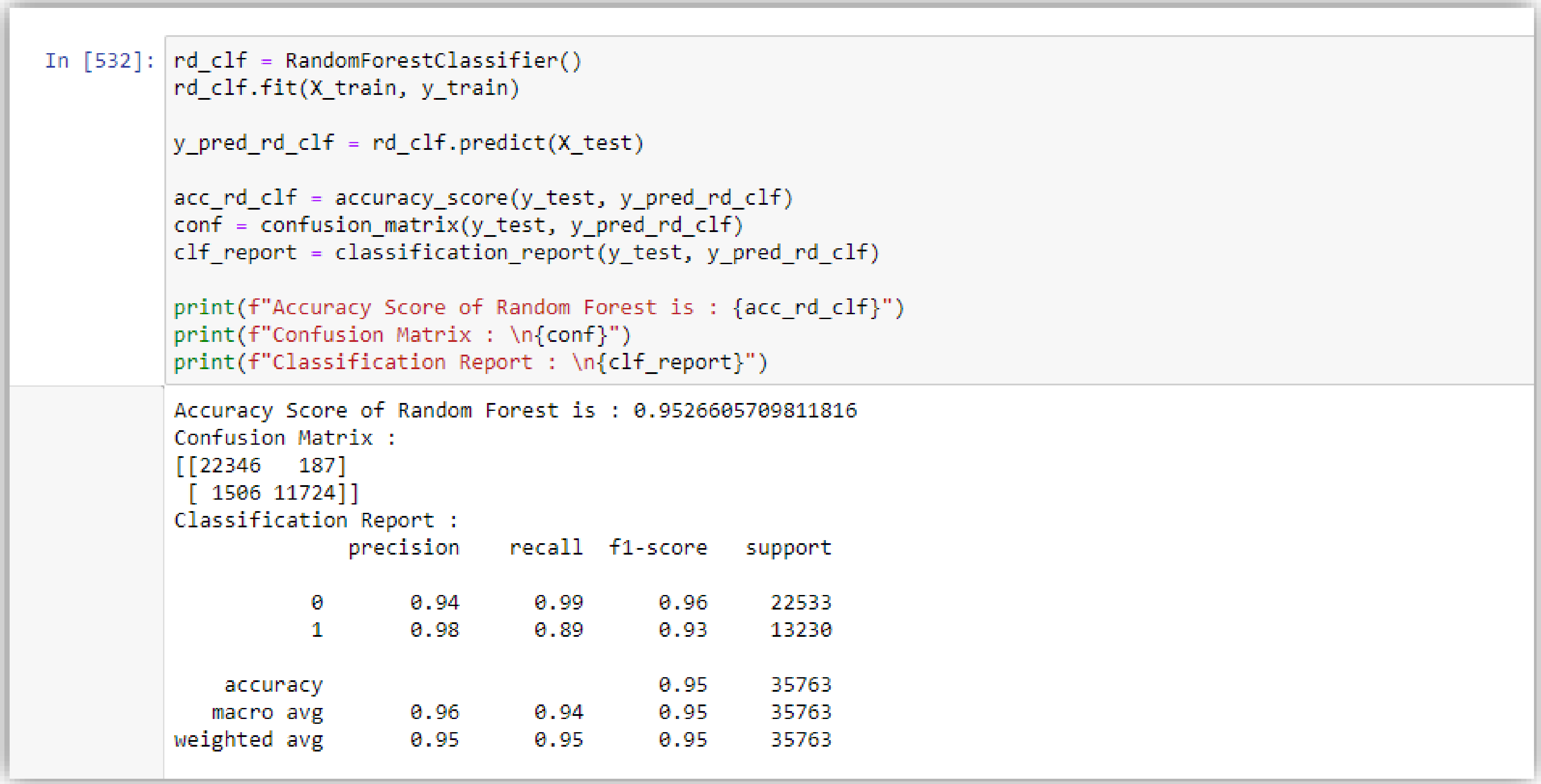


In [531]:

```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

y_pred_dtc = dtc.predict(X_test)

acc_dtc = accuracy_score(y_test, y_pred_dtc)
conf = confusion_matrix(y_test, y_pred_dtc)
clf_report = classification_report(y_test, y_pred_dtc)

print(f"Accuracy Score of Decision Tree is : {acc_dtc}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Decision Tree is : 0.9416156362721249
Confusion Matrix :
[[21490  1043]
 [ 1045 12185]]
Classification Report :
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     22533
           1       0.92      0.92      0.92     13230

    accuracy                           0.94     35763
   macro avg       0.94      0.94      0.94     35763
weighted avg       0.94      0.94      0.94     35763
```

RANDOM FOREST

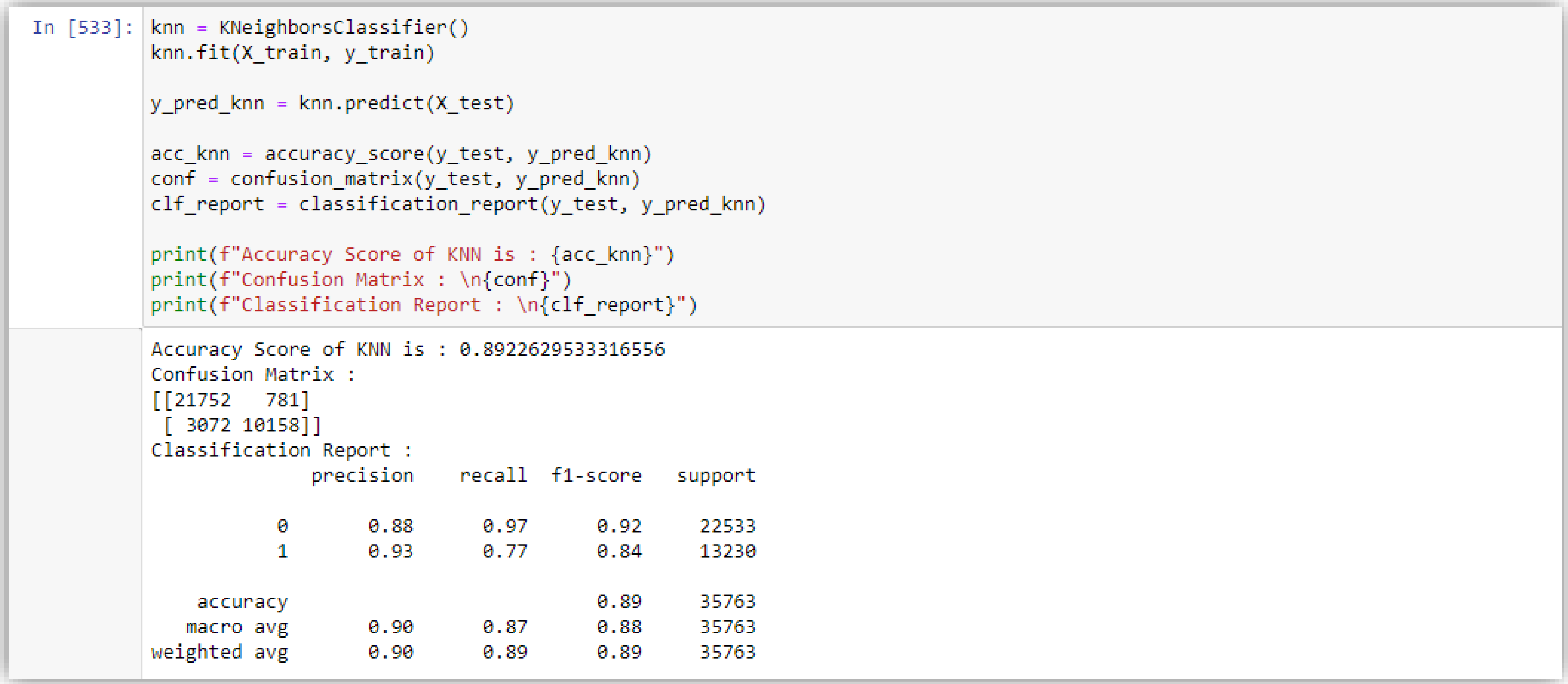


In [532]:

```
rd_clf = RandomForestClassifier()
rd_clf.fit(X_train, y_train)

y_pred_rd_clf = rd_clf.predict(X_test)

acc_rd_clf = accuracy_score(y_test, y_pred_rd_clf)
conf = confusion_matrix(y_test, y_pred_rd_clf)
clf_report = classification_report(y_test, y_pred_rd_clf)

print(f"Accuracy Score of Random Forest is : {acc_rd_clf}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Random Forest is : 0.9526605709811816
Confusion Matrix :
[[22346   187]
 [ 1506 11724]]
Classification Report :
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     22533
           1       0.98      0.89      0.93     13230

    accuracy                           0.95     35763
   macro avg       0.96      0.94      0.95     35763
weighted avg       0.95      0.95      0.95     35763
```

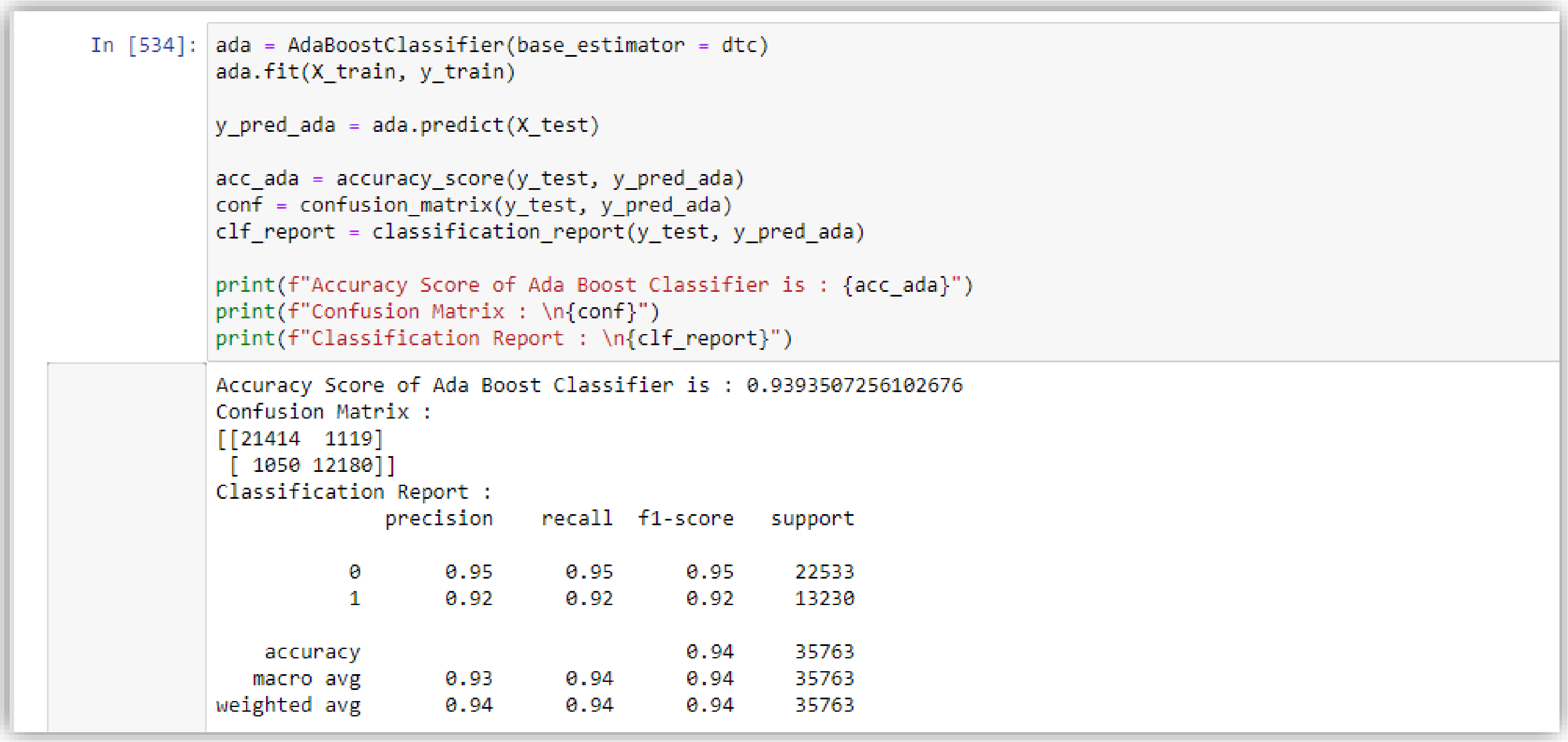
KNN



```python
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

acc_knn = accuracy_score(y_test, y_pred_knn)
conf = confusion_matrix(y_test, y_pred_knn)
clf_report = classification_report(y_test, y_pred_knn)

print(f"Accuracy Score of KNN is : {acc_knn}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of KNN is : 0.8922629533316556
Confusion Matrix : 
[[21752   781]
 [ 3072 10158]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.88      0.97      0.92     22533
           1       0.93      0.77      0.84     13230

    accuracy                           0.89     35763
   macro avg       0.90      0.87      0.88     35763
weighted avg       0.90      0.89      0.89     35763
```

ADA BOOST CLASSIFIER

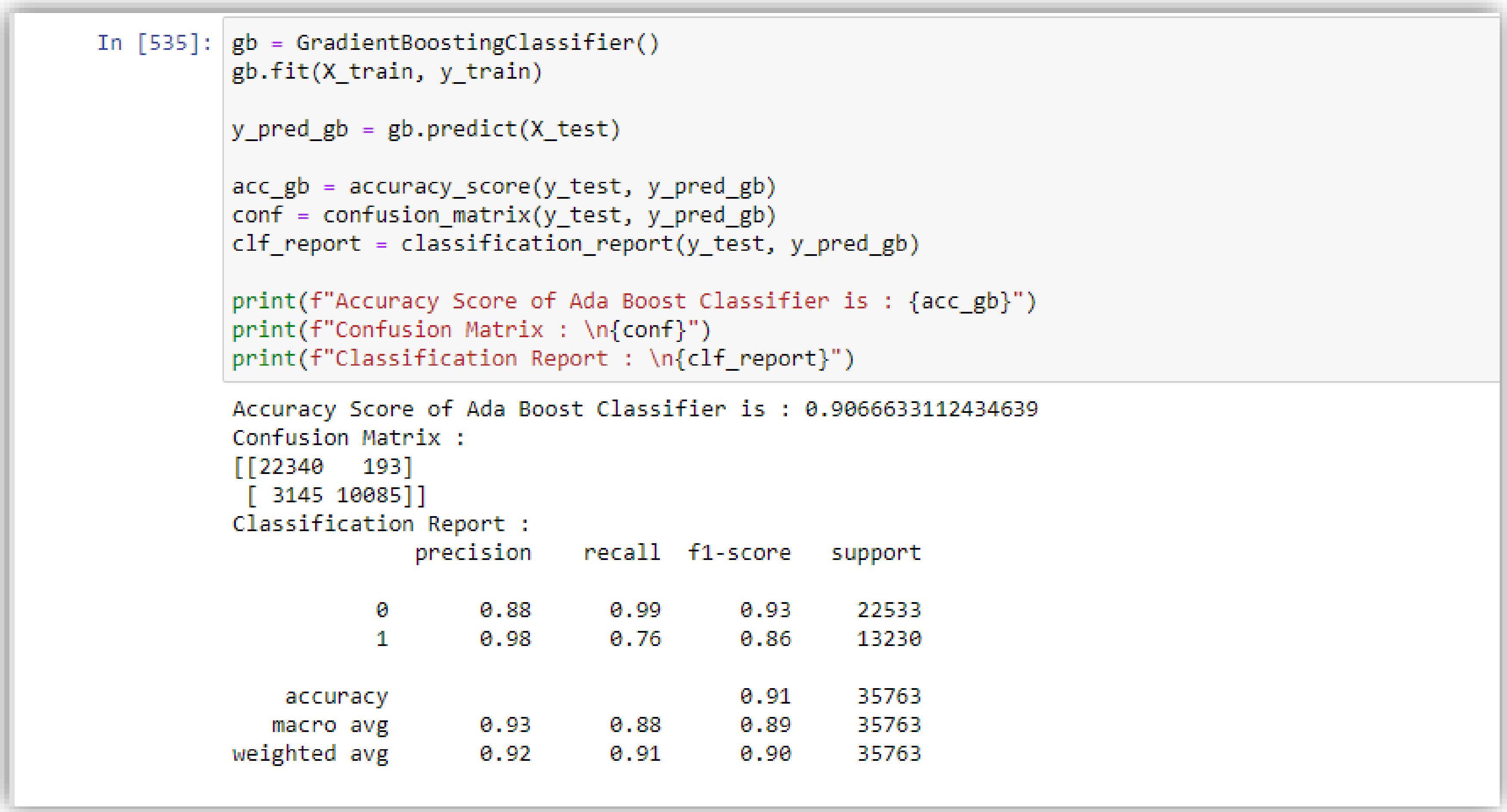


In [534]:

```
ada = AdaBoostClassifier(base_estimator = dtc)
ada.fit(X_train, y_train)

y_pred_ada = ada.predict(X_test)

acc_ada = accuracy_score(y_test, y_pred_ada)
conf = confusion_matrix(y_test, y_pred_ada)
clf_report = classification_report(y_test, y_pred_ada)

print(f"Accuracy Score of Ada Boost Classifier is : {acc_ada}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9393507256102676
Confusion Matrix : 
[[21414  1119]
 [ 1050 12180]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     22533
           1       0.92      0.92      0.92     13230

    accuracy                           0.94     35763
   macro avg       0.93      0.94      0.94     35763
weighted avg       0.94      0.94      0.94     35763
```

GRADIENT BOOSTING CLASSIFIER

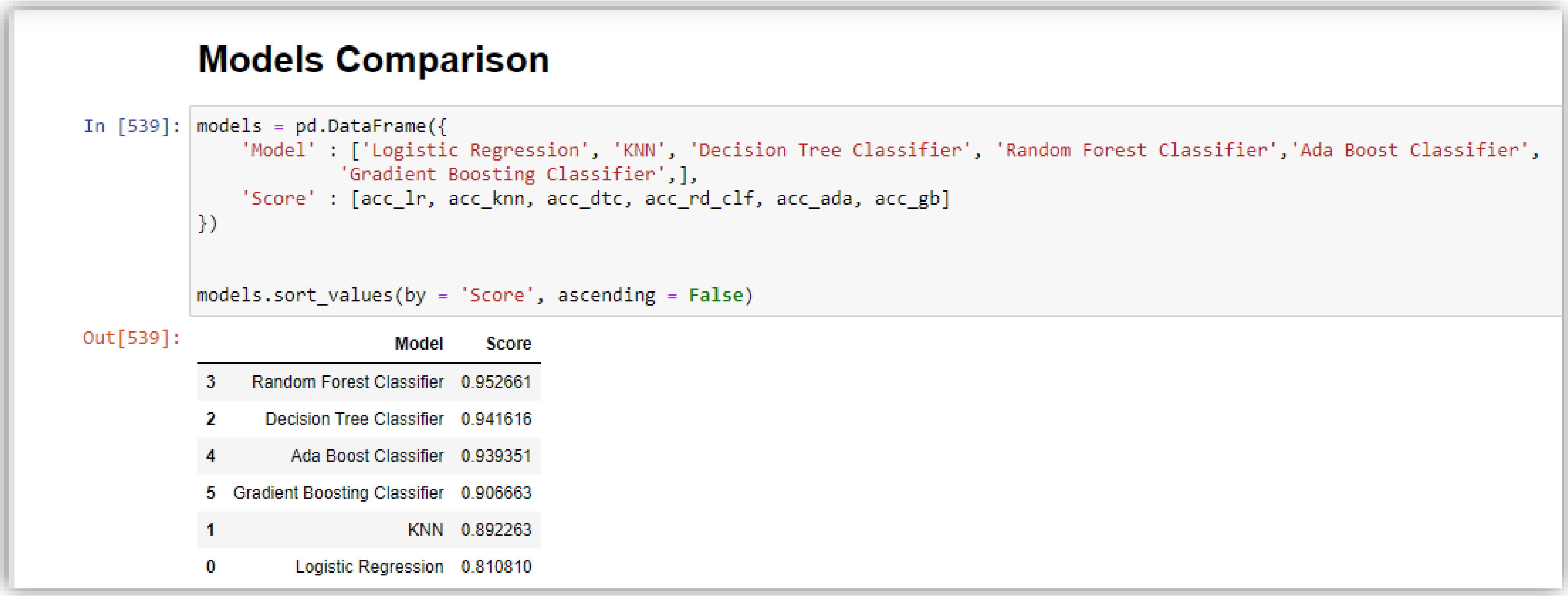


In [535]:

```
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)

y_pred_gb = gb.predict(X_test)

acc_gb = accuracy_score(y_test, y_pred_gb)
conf = confusion_matrix(y_test, y_pred_gb)
clf_report = classification_report(y_test, y_pred_gb)

print(f"Accuracy Score of Ada Boost Classifier is : {acc_gb}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9066633112434639
Confusion Matrix :
[[22340   193]
 [ 3145 10085]]
Classification Report :
              precision    recall  f1-score   support

           0       0.88      0.99      0.93     22533
           1       0.98      0.76      0.86     13230

    accuracy                           0.91     35763
   macro avg       0.93      0.88      0.89     35763
weighted avg       0.92      0.91      0.90     35763
```

MODELS COMPARISON



Models Comparison

In [539]:

```
models = pd.DataFrame({
    'Model' : ['Logistic Regression', 'KNN', 'Decision Tree Classifier', 'Random Forest Classifier','Ada Boost Classifier',
             'Gradient Boosting Classifier',],
    'Score' : [acc_lr, acc_knn, acc_dtc, acc_rd_clf, acc_ada, acc_gb]
})


models.sort_values(by = 'Score', ascending = False)
```

Out[539]:

	Model	Score
3	Random Forest Classifier	0.952661
2	Decision Tree Classifier	0.941616
4	Ada Boost Classifier	0.939351
5	Gradient Boosting Classifier	0.906663
1	KNN	0.892263
0	Logistic Regression	0.810810

CONCLUSIONI

Come possiamo vedere dalla slide precedente, l'algoritmo Random Forest ha prodotto la migliore accuratezza.

Questo punteggio è buono e ideale in quanto riesce a predire con sicurezza la prenotazione di una camera in un hotel .