Title: Predicting Flight Prices: A ML Approach

Dataset : https://www.kaggle.com/datasets/shubh...

1. **Dataset Overview**
   - Loading the data:

```
import pandas as pd
import numpy as np
import sklearn as sk

df = pd.read_csv('/Users/rd/coding_stuff/Projects/Fligh
ts Price prediction/dataset/Clean_Dataset.csv')
```

   - Exploring key features:

```
df.airline.value_counts()
df.source_city.value_counts()
df.destination_city.value_counts()
df.departure_time.value_counts()
df.arrival_time.value_counts()
df.stops.value_counts()
df['class'].value_counts()
```

2. **Data Preprocessing**
   - Removing unnecessary columns:

```
df = df.drop('Unnamed: 0', axis=1)
df = df.drop('flight', axis=1)
```

   - Encoding categorical variables:

```
df['class'] = df['class'].apply(lambda x: 1 if x=='Busi
ness' else 0)
df.stops = pd.factorize(df.stops)[0]
```

   - One-hot encoding for airlines, cities, and time slots:

```
df = df.join(pd.get_dummies(df.airline, dtype=int, pref
ix='airline')).drop('airline', axis=1)
```

```
df = df.join(pd.get_dummies(df.source_city,dtype=int, p
refix='source')).drop('source_city', axis=1)
df = df.join(pd.get_dummies(df.destination_city,dtype=i
nt, prefix='destination')).drop('destination_city', axi
s=1)
df = df.join(pd.get_dummies(df.arrival_time,dtype=int,
prefix='arrival')).drop('arrival_time', axis=1)
df = df.join(pd.get_dummies(df.departure_time,dtype=in
t, prefix='departure')).drop('departure_time', axis=1)
```

3. **Model Selection and Training**
    - Choosing Random Forest Regressor
    - Splitting data and training the model:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

x,y = df.drop('price', axis=1), df.price
x_train, x_test, y_train, y_test = train_test_split(x,
y, test_size=0.2)

reg = RandomForestRegressor(n_jobs=-1)
reg.fit(x_train, y_train)
```

4. **Model Evaluation**
    - Calculating performance metrics:

```
from sklearn.metrics import mean_absolute_error, mean_s
quared_error, r2_score

y_pred = reg.predict(x_test)
print('R2', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
print('MSE', mean_squared_error(y_test, y_pred))
```

    - Visualising predicted vs actual prices:

```
import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred)
plt.xlabel('Actual Flight Price')
plt.ylabel('Predicted Flight price')
plt.title('Predicted vs Actual')
plt.show()
```

5. **Feature Importance Analysis**

- Identifying and visualizing important features:

```
importances = dict(zip(reg.feature_names_in_, reg.featu
re_importances_))
sorted_importances = sorted(importances.items(), key=la
mbda x: x[1], reverse=True)

plt.figure(figsize=(15,6))
plt.bar([x[0] for x in sorted_importances[:10]], [x[1]
for x in sorted_importances[:10]])
plt.show()
```

6. **Hyperparameter Tuning**
   - Using GridSearchCV for finding optimal parameters:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_leaf': [2, 5, 10],
    'max_features': ['auto', 'sqrt']
}

grid_search = GridSearchCV(reg, param_grid, cv=5)
grid_search.fit(x_train, y_train)

best_params = grid_search.best_params_
```