# ⍰Aviachipta narxini bashorat qilish:

Albatta hech kim aviachipta narxi qancha bo'linishini oldindan aniq aytolmaydi. Ammo biz berilgan ma`lumotlarga ko'ra aviachipta narxini bashorat qilishga harakat qilamiz.

## ⍰Kerakli kutubxonalar:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
import html5lib
%matplotlib inline
```

## Datasetni chaqirib olamiz(train_data):

```
df = pd.read_csv("/content/train_data.csv", index_col=0)
df.head()
```

| | airline | flight | source_city | departure_time | stops | arival_time | destination_city | class | duration | days_left | p |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **id** | | | | | | | | | | | |
| **1** | Vistara | UK-810 | Bangalore | Early_Morning | one | Night | Mumbai | Economy | 14.25 | 21 | 7 |
| **2** | SpiceJet | SG-5094 | Hyderabad | Evening | zero | Night | Kolkata | Economy | 1.75 | 7 | 5 |
| **3** | Vistara | UK-846 | Bangalore | Morning | one | Evening | Delhi | Business | 9.58 | 5 | 60 |
| **4** | Vistara | UK-706 | Kolkata | Morning | one | Evening | Hyderabad | Economy | 6.75 | 28 | 5 |
| **5** | Indigo | 6E-5394 | Chennai | Early_Morning | zero | Morning | Mumbai | Economy | 2.00 | 4 | 10 |

Ushbu ma’lumotlar to‘plami quyidagilarni o'z ichiga oladi:

Tarkib:

ID: Ketma - ketlik uchun qo'yilgan sonlar.

Airline: Parvoz qilingan aviakompaniya nomi.

Flight: Parvoz qilingan ID raqami.

Source_city: Parvoz qaysi shahardan boshlanishi.

Departure_time: Samolyotning ketish vaqti.

Stop: Parvoz davomida to'xtashlar soni.

Arrival_time: Qaytish vaqti.

Destination_city: Qaysi davlatga parvoz qilinayotgani.

Class: Foydalanuvchi samolyotning qaysi klass turidan foydalanib uchganligi.

Duration: Parvoz davomiyligi.

Days_left: Samolyotning qancha vaqtdan keyin qaytishi

## Tarkibni tekshirib olamiz:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 1 to 20000
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   airline           20000 non-null  object
 1   flight            20000 non-null  object
 2   source_city       20000 non-null  object
 3   departure_time    20000 non-null  object
 4   stops             20000 non-null  object
 5   arrival_time      20000 non-null  object
 6   destination_city  20000 non-null  object
 7   class             20000 non-null  object
 8   duration          20000 non-null  float64
 9   days_left         20000 non-null  int64
 10  price             20000 non-null  int64
dtypes: float64(1), int64(2), object(8)
memory usage: 1.8+ MB
```

## Kategoriyaga bo`lingan ustunlar soni:

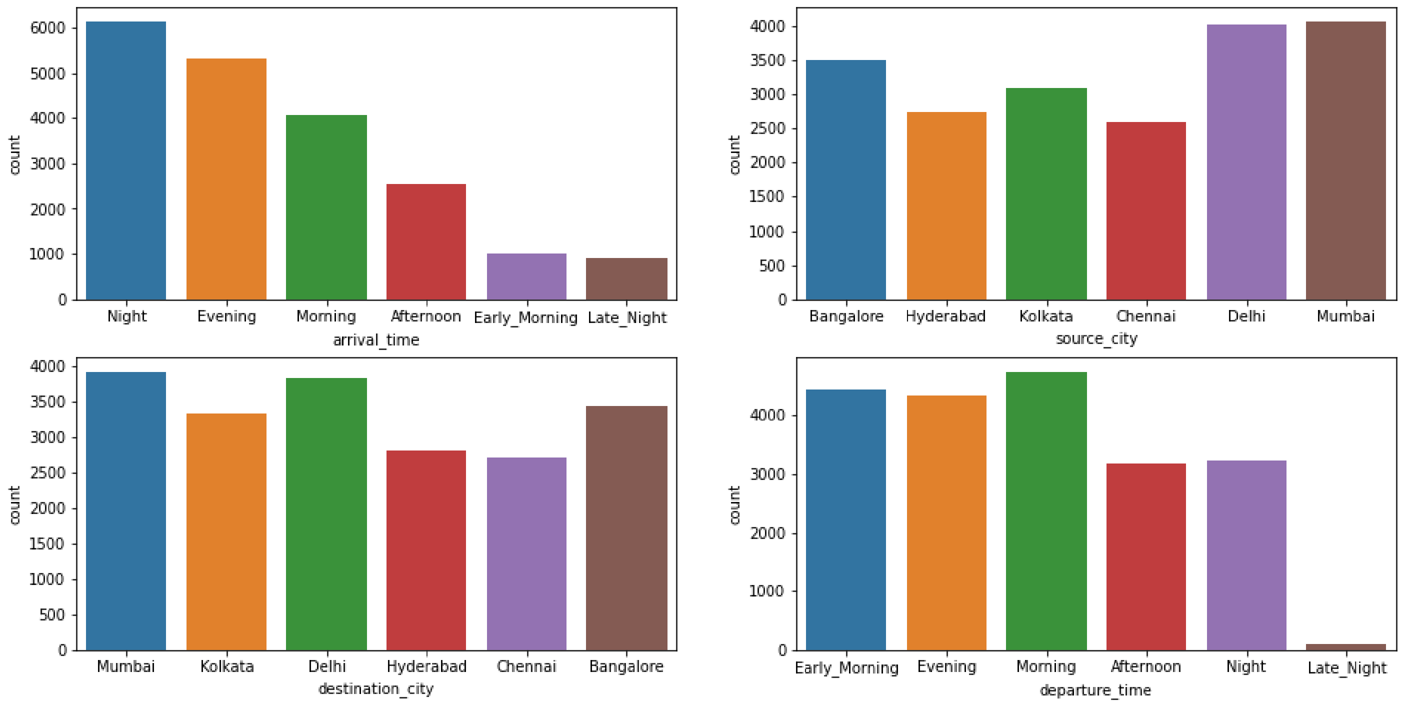
```
print(f'Toxtashlar: \n\n{df.stops.value_counts()}\n')
print(f'Qaytish vaqti:\n\n {df.arrival_time.value_counts()}\n')
print(f"AviaComp: \n\n{df.airline.value_counts()}\n")
print(f"Qaysi davlatga parvoz qilinayotgani:\n\n {df.destination_city.value_counts()}\r
print(f"Parvoz qaysi shahardan boshlanishi: \n\n {df.source_city.value_counts()}\n")
```

```
Toxtashlar:


one            16666
zero            2440
two_or_more      894
Name: stops, dtype: int64


Qaytish vaqti:
 Night            6142
Evening          5316
Morning          4086
Afternoon        2536
Early_Morning    1004
Late_Night        916
Name: arrival_time, dtype: int64

AviaComp:

Vistara      8535
Air_India    5371
Indigo       2924
GO_FIRST     1508
AirAsia      1056
SpiceJet      606
Name: airline, dtype: int64

Qaysi davlatga parvoz qilinayotgani:

 Mumbai       3918
Delhi        3825
Bangalore    3425
Kolkata      3323
Hyderabad    2814
Chennai      2695
Name: destination_city, dtype: int64

Parvoz qaysi shahardan boshlanishi:

 Mumbai       4068
Delhi        4022
Bangalore    3486
Kolkata      3092
Hyderabad    2748
Chennai      2584
Name: source_city, dtype: int64
```

Vizual ko`rinishda :

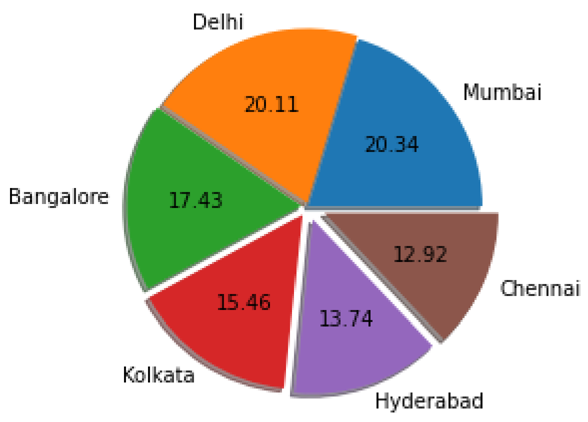
```
fig, ax = plt.subplots(2,2, figsize=(16,8))
```

```
sns.countplot(ax = ax[0,0], data=df, x='arrival_time')
sns.countplot(ax = ax[0,1], data=df, x = 'source_city')
sns.countplot(ax = ax[1,0], data=df, x = 'destination_city')
sns.countplot(ax = ax[1,1], data=df, x = 'departure_time')

plt.show()
```



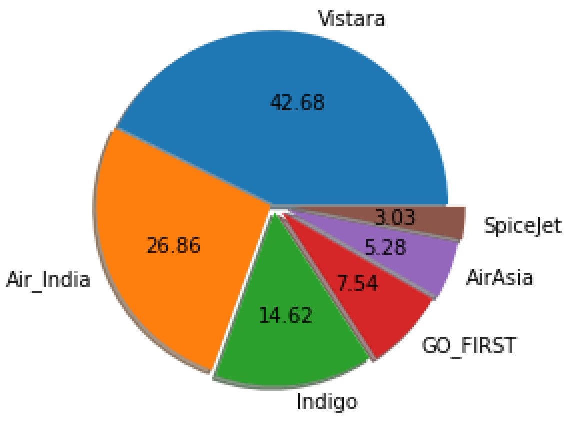
source_city

```
list_sc=list(dict(df.source_city.value_counts()).keys())
data_sc = list(df.source_city.value_counts())


from pandas.core.arrays.interval import value_counts
plt.pie(
    data_sc,
    labels = list_sc,
    explode = [0, 0.02, 0.04, 0.06, 0.08, 0.1],
    autopct='%.2f',
    shadow = True)
plt.show()
```



```
airline
```

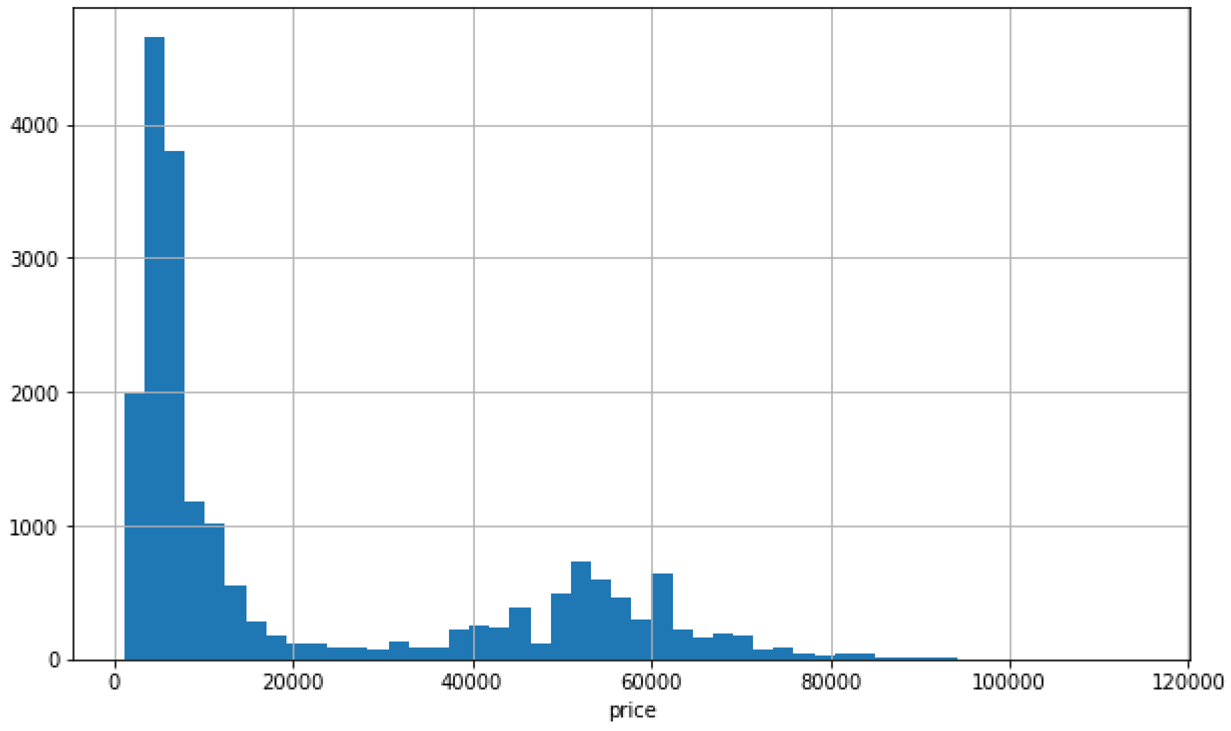
```
list_al =list(dict(df.airline.value_counts()).keys())
data_al = list(df.airline.value_counts())

plt.pie(
    data_al,
    labels = list_al,
    explode = [0, 0.02, 0.04, 0.06, 0.08, 0.1],
    autopct='%.2f',
    shadow = True)

plt.show()
```



Ma`lumotlarni tozalash:

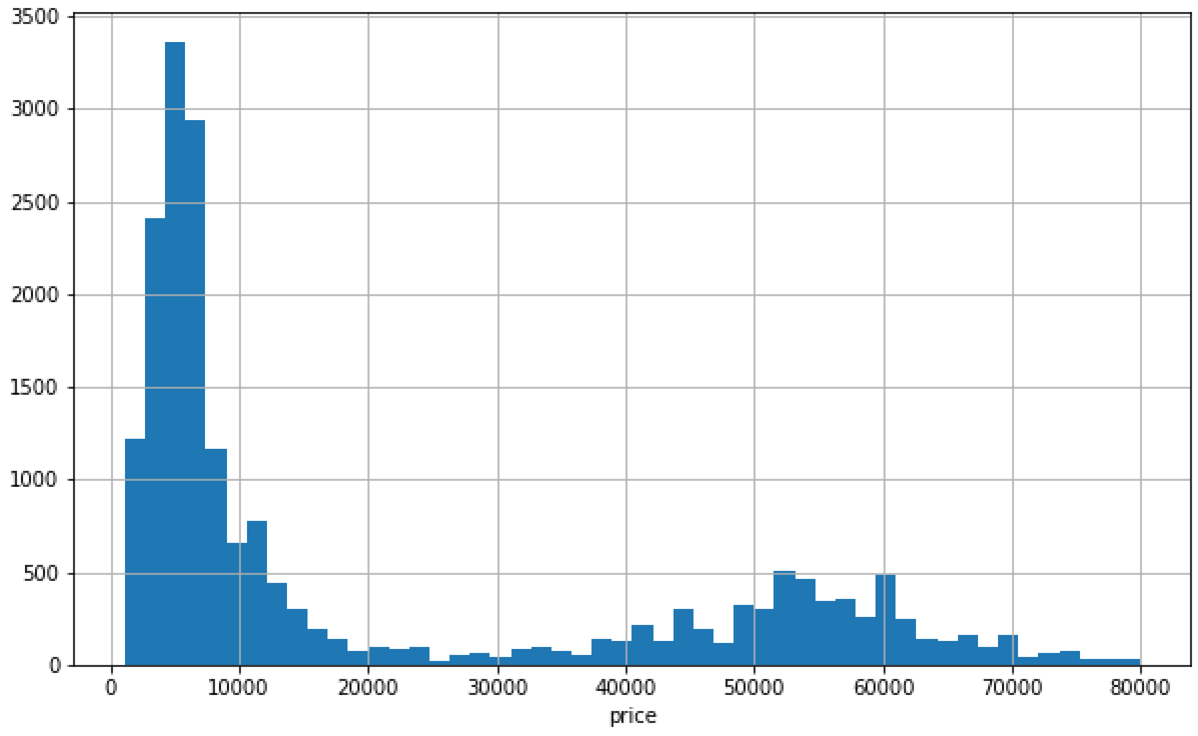
## `price` ustunini tekshiramiz.

```
plt.figure(figsize=(10,6))
plt.hist(df['price'], bins=50, histtype='bar')
plt.xlabel('price')
plt.grid()
plt.show()
```



```python
df = df[df['price']<80000]
```

```python
plt.figure(figsize=(10,6))
plt.hist(df['price'], bins=50, histtype='bar')
plt.xlabel('price')
plt.grid()
plt.show()
```



Indexlarni yangilaymiz:

```python
df.index = list(range(len(df)))
```

## datasetni ni ikki qismga ajratib olamiz. train va test setga. Bu uchun `StratifiedShuffleSplit` dan foydalanamiz

```
df['price_cat'] = pd.cut(df['price'], bins = [0, 5000, 10000, 20000, 50000, 80000, np.i

from sklearn.model_selection import StratifiedShuffleSplit
stratified_split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in stratified_split.split(df, df['price_cat']):
    strat_train_set = df.loc[train_index]
    strat_test_set = df.loc[test_index]

strat_train_set.drop('price_cat', axis=1, inplace=True)
strat_test_set.drop('price_cat', axis=1, inplace=True)
```

```
df_train = strat_train_set.copy()
df_test = strat_test_set.copy()
```

### Matnli ustunlarning qiymatlarini raqamlarga o`tkazib olamiz va korrelyatsiyani tekshirib koramiz.

```
# matnli ustunlar
df_train_cat = df_train[['airline','flight','source_city','departure_time','stops','arr
# raqamli ustunlar
df_train_num = df_train[['duration','days_left']]
```

## OrdinalEncoder

```
from sklearn.preprocessing import OrdinalEncoder
cat_encoder = OrdinalEncoder()
df_cat_encoded = cat_encoder.fit_transform(df_train_cat)
df_cat_encoded
```

```
array([[5.000e+00, 1.186e+03, 1.000e+00, ..., 5.000e+00, 5.000e+00,
        0.000e+00],
       [1.000e+00, 6.910e+02, 2.000e+00, ..., 0.000e+00, 4.000e+00,
        1.000e+00],
       [1.000e+00, 5.310e+02, 2.000e+00, ..., 2.000e+00, 4.000e+00,
        1.000e+00],
       ...,
       [2.000e+00, 8.540e+02, 4.000e+00, ..., 2.000e+00, 2.000e+00,
        1.000e+00],
       [1.000e+00, 5.360e+02, 4.000e+00, ..., 2.000e+00, 3.000e+00,
        0.000e+00],
       [5.000e+00, 1.234e+03, 2.000e+00, ..., 1.000e+00, 0.000e+00,
        0.000e+00]])
```

## Xuddi shu `df_cat` ga `price`, `days_left`,`duration` ustunlarini qo`shib korrelyatsiyani tekshiramiz

```
df_cat = pd.DataFrame(df_cat_encoded,columns = df_train_cat.columns, index = df_train_c
df_cat['price'] = df_train['price']
df_cat['days_left'] = df_train['days_left']
df_cat['duration'] = df_train['duration']
df_cat.head()
```

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | price | days_left | dura |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **3792** | 5.0 | 1186.0 | 1.0 | 4.0 | 0.0 | 5.0 | 5.0 | 0.0 | 69713 | 43 | 1 |
| **14310** | 1.0 | 691.0 | 2.0 | 4.0 | 0.0 | 0.0 | 4.0 | 1.0 | 4559 | 24 | 2 |
| **369** | 1.0 | 531.0 | 2.0 | 4.0 | 0.0 | 2.0 | 4.0 | 1.0 | 17641 | 9 | |
| **1397** | 1.0 | 566.0 | 2.0 | 4.0 | 0.0 | 5.0 | 5.0 | 0.0 | 53164 | 43 | 1 |
| **2914** | 1.0 | 672.0 | 5.0 | 1.0 | 0.0 | 2.0 | 4.0 | 0.0 | 49725 | 7 | 1 |

```
df_cat.corrwith(df_cat['price']).sort_values(ascending=False)
```

```
price               1.000000
flight              0.291334
airline             0.226475
duration            0.215972
departure_time      0.059233
arrival_time        0.042076
source_city         0.008464
destination_city    0.006042
days_left          -0.091782
stops              -0.201906
class              -0.942021
dtype: float64
```

## Grafik ko`rinishda chiqaramiz:

```
corr_matrix = df_cat.corr().abs()
corr_matrix.style.background_gradient(cmap='coolwarm')
```

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class |
|---|---|---|---|---|---|---|---|---|
| **airline** | 1.000000 | 0.639660 | 0.023320 | 0.053067 | 0.027900 | 0.030755 | 0.037068 | 0.162699 |
| **flight** | 0.639660 | 1.000000 | 0.013321 | 0.065870 | 0.112804 | 0.064710 | 0.037579 | 0.244361 |
| **source_city** | 0.023320 | 0.013321 | 1.000000 | 0.013545 | 0.002085 | 0.036516 | 0.234042 | 0.005826 |
| **departure_time** | 0.053067 | 0.065870 | 0.013545 | 1.000000 | 0.017414 | 0.035813 | 0.008150 | 0.057898 |
| **stops** | 0.027900 | 0.112804 | 0.002085 | 0.017414 | 1.000000 | 0.005930 | 0.010782 | 0.092243 |
| **arrival_time** | 0.030755 | 0.064710 | 0.036516 | 0.035813 | 0.005930 | 1.000000 | 0.026371 | 0.032210 |
| **destination_city** | 0.037068 | 0.037579 | 0.234042 | 0.008150 | 0.010782 | 0.026371 | 1.000000 | 0.002138 |
| **class** | 0.162699 | 0.244361 | 0.005826 | 0.057898 | 0.092243 | 0.032210 | 0.002138 | 1.000000 |

|  | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class |
|---|---|---|---|---|---|---|---|---|
| **price** | 0.226475 | 0.291334 | 0.008464 | 0.059233 | 0.201906 | 0.042076 | 0.006042 | 0.942021 |
| **days_left** | 0.022624 | 0.001512 | 0.001436 | 0.008358 | 0.024714 | 0.002032 | 0.013927 | 0.014580 |
| **duration** | 0.011130 | 0.197053 | 0.010330 | 0.090829 | 0.482574 | 0.014237 | 0.006742 | 0.142789 |

```
df_train.head(3)
```

|  | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left |
|---|---|---|---|---|---|---|---|---|---|---|
| **3792** | Vistara | UK-836 | Chennai | Morning | one | Night | Mumbai | Business | 11.25 | 43 |
| **14310** | Air_India | AI-889 | Delhi | Morning | one | Afternoon | Kolkata | Economy | 27.58 | 24 |
| **369** | Air_India | AI-409 | Delhi | Morning | one | Evening | Kolkata | Economy | 7.00 | 9 |

```
# matnli ustunlar
X_train_cat = df_train[['airline','flight','stops','class','source_city','departure_tim
# raqamli ustunlar
X_train_num = df_train[['duration','days_left']]
# ylabel
y_train = df_train[['price']]
# xlabel
X_train = df_train.drop('price', axis=1)
```

## Pipline yaratib olamiz

Raqamli ustunlar bilan ishlaydigan pipline:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
          ('std_scaler', StandardScaler())
])
```

## Matnli ustunlar bilan ham ishlaydigan umumiy `pipline` yozamiz

full_pipline:

```
from sklearn.compose import ColumnTransformer

num_attribs = list(X_train_num)
cat_attribs = list(X_train_cat)

full_pipeline = ColumnTransformer([
    ('cat', OrdinalEncoder(), cat_attribs),
    ('num', num_pipeline, num_attribs),
])
```

`X_train` ni `full_pipeline` dan utkazib olamiz

```
X_prepared = full_pipeline.fit_transform(X_train)
X_prepared
```

```
array([[ 5.00000000e+00,  1.18600000e+03,  0.00000000e+00, ...,
         5.00000000e+00, -1.30422093e-01,  1.25132612e+00],
       [ 1.00000000e+00,  6.91000000e+02,  0.00000000e+00, ...,
         4.00000000e+00,  2.14494186e+00, -1.46288775e-01],
       [ 1.00000000e+00,  5.31000000e+02,  0.00000000e+00, ...,
         4.00000000e+00, -7.22601934e-01, -1.24966895e+00],
       ...,
       [ 2.00000000e+00,  8.54000000e+02,  0.00000000e+00, ...,
         2.00000000e+00, -3.63113936e-01,  1.25132612e+00],
       [ 1.00000000e+00,  5.36000000e+02,  0.00000000e+00, ...,
         3.00000000e+00,  2.45844884e+00,  1.10420876e+00],
       [ 5.00000000e+00,  1.23400000e+03,  0.00000000e+00, ...,
         0.00000000e+00,  4.03236446e-01, -1.10255160e+00]])
```

## Ma`lumotlar ML uchun tayyor

### Machine Learning

## `Linear Regression` - Chiziqli regressiya

sklearn tarkibidagi `LinearRegression` klassidan yangi model yaratamiz.

```
from sklearn.linear_model import LinearRegression
LR_model = LinearRegression()

LR_model.fit(X_prepared, y_train)
```

```
LinearRegression()
```

## RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor
RF_model = RandomForestRegressor()

RF_model.fit(X_prepared, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples,), for example using ravel().
  after removing the cwd from sys.path.
```

```
RandomForestRegressor()
```

##Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
Tree_model = DecisionTreeRegressor()

Tree_model.fit(X_prepared, y_train)
```

```
DecisionTreeRegressor()
```

## Modelni tekshirib ko`ramiz.

##Bunda test qilish uchun ajratganimiz `df_test` dan foydalanamiz.

```
df_test.head(3)
```

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days |
|---|---|---|---|---|---|---|---|---|---|---|
| **10796** | Indigo | 6E-929 | Chennai | Afternoon | one | Night | Kolkata | Economy | 8.83 | |
| **17971** | GO_FIRST | G8-791 | Bangalore | Evening | one | Early_Morning | Mumbai | Economy | 11.75 | |
| **2960** | Vistara | UK-849 | Mumbai | Evening | zero | Evening | Bangalore | Economy | 1.75 | |

```
y_test = df_test[['price']]
X_test = df_test.drop('price', axis=1)
```

`X_test` ni full_piplinedan o`tkazvolamiz:

```
y_test['price'].values
```

```
array([4079, 5177, 2074, ..., 4764, 5206, 3272])
```

```
# X_test ni full_piplinedan o`tkazvolamiz:
X_test_prepared = full_pipeline.fit_transform(X_test)

y_predict = RF_model.predict(X_test_prepared)

y_predict
pd.DataFrame({'Bashorat':y_predict, 'Asl qiymat': y_test['price'].values})
```

| | Bashorat | Asl qiymat |
|---|---|---|
| **0** | 3543.03 | 4079 |
| **1** | 5736.33 | 5177 |
| **2** | 2097.00 | 2074 |
| **3** | 3353.83 | 3393 |
| **4** | 4385.42 | 2622 |
| **...** | ... | ... |
| **3968** | 5177.80 | 3728 |
| **3969** | 5200.37 | 4149 |

|  | Bashorat | Asl qiymat |
|---|---|---|
| **3970** | 5634.67 | 4764 |
| **3971** | 4799.21 | 5206 |
| **3972** | 4412.48 | 3272 |

3973 rows × 2 columns

## Xatolikni tekshiramiz:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y_test.values, y_predict)
rmse = np.sqrt(mean_squared_error(y_test.values, y_predict))
print(f"MAE = {mae}")
print(f'RMSE = {rmse}')
```

```
MAE = 3425.661391248067
RMSE = 5888.9877151899045
```

##Cross-Validation

```
X = df.drop("price", axis=1)
y = df["price"].copy()

X_prepared = full_pipeline.fit_transform(X)
```

Validation natijalarini ko'rsatish uchun sodda funksiya yasab olamiz

```
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Std.dev:", scores.std())
```

##Random Forest

```
from sklearn.model_selection import cross_val_score

# predict
scores = cross_val_score(RF_model, X_prepared, y, scoring="neg_mean_squared_error", cv=
RF_rmse_scores = np.sqrt(-scores)

# xatolik
display_scores(RF_rmse_scores)
```

```
Scores: [3031.04484158 2646.14841863 2974.75840565 2699.77024702 2838.80238396
 2825.62202267 2789.253513   2893.15137428 2903.98375543 2814.18053179]
Mean: 2841.671549400101
Std.dev: 110.56303124391725
```

#LinearRegression

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(LR_model, X_prepared, y, scoring="neg_mean_squared_error", cv=
LR_rmse_scores = np.sqrt(-scores)

display_scores(LR_rmse_scores)
```

```
Scores: [6606.93280356 6248.21268504 6470.33750377 6287.70609637 6548.8722286
 6552.03795535 6443.69176351 6568.93018082 6571.43923125 6606.66053347]
Mean: 6490.482098174364
Std.dev: 122.21808422364852
```

## Decision-Tree

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(Tree_model, X_prepared, y, scoring="neg_mean_squared_error", c
DT_rmse_scores = np.sqrt(-scores)

display_scores(DT_rmse_scores)
```

```
Scores: [4150.12237814 3641.51916484 4011.52750966 3581.34963138 3739.08501094
 3644.77932567 3883.80643732 3698.6653911  3748.02124722 3669.21259379]
Mean: 3776.808869006233
Std.dev: 173.1412406539246
```

### Amaliyotimiz uchun eng yaxshi model `RF_model` ekan.

## modelni saqlaymiz:

```
import pickle

filename = 'RF_model_finally.pkl' # faylga istalgan nom beramiz
with open(filename, 'wb') as file:
    pickle.dump(RF_model, file)
```

### ✔Endi bizdan bashorat qilinishi so'ralgan `test_data` dagi malumotlarga ko'ra `price` ni aniqlaymiz:

```
df_test_new = pd.read_csv('/content/test_data.csv',index_col=0)
df_test_new.head()
```

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left |
|---|---|---|---|---|---|---|---|---|---|---|

| id | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left |
|---|---|---|---|---|---|---|---|---|---|---|
| id | | | | | | | | | | |
| 1 | Air_India | AI-765 | Kolkata | Evening | one | Night | Delhi | Business | 28.25 | 2 |
| 2 | Vistara | UK-747 | Delhi | Early_Morning | one | Night | Mumbai | Business | 13.83 | 34 |
| 3 | Air_India | AI-570 | Mumbai | Early_Morning | zero | Early_Morning | Chennai | Business | 2.00 | 30 |
| 4 | AirAsia | I5-974 | Hyderabad | Night | one | Late_Night | Delhi | Economy | 5.17 | 26 |
| 5 | Air_India | AI-770 | Kolkata | Night | one | Afternoon | Mumbai | Economy | 16.33 | 35 |

## `full_pipline` dan o'tkazib olamiz va `RF_model` ga uzatamiz:

```
df_test_prepared = full_pipeline.fit_transform(df_test_new)
df_test_prepared
```

```
array([[ 1.00000000e+00,  4.52000000e+02,  0.00000000e+00, ...,
         2.00000000e+00,  2.17930407e+00, -1.75411081e+00],
       [ 5.00000000e+00,  8.34000000e+02,  0.00000000e+00, ...,
         5.00000000e+00,  2.05480508e-01,  5.83184361e-01],
       [ 1.00000000e+00,  4.01000000e+02,  2.00000000e+00, ...,
         1.00000000e+00, -1.41382134e+00,  2.91022465e-01],
       ...,
       [ 1.00000000e+00,  3.44000000e+02,  0.00000000e+00, ...,
         5.00000000e+00, -1.58622867e-01,  8.02305784e-01],
       [ 1.00000000e+00,  4.32000000e+02,  0.00000000e+00, ...,
         3.00000000e+00,  1.37040024e-01,  8.75346258e-01],
       [ 5.00000000e+00,  8.39000000e+02,  0.00000000e+00, ...,
         2.00000000e+00,  5.13462686e-01,  1.67879147e+00]])
```

## Bashorat:

```
df_pred = RF_model.predict(df_test_prepared)
df_pred
```

```
array([53866.19, 45926.32, 25289.  , ..., 52007.38, 52595.58, 47848.7 ])
```

```
answer_df = pd.read_csv("/content/sample_solution.csv",index_col=0)
answer_df['price'] = df_pred.astype(int)
```

```
answer_df
```

| | price |
|---|---|
| id | |

| | price |
|---|---|
| **id** | |
| **1** | 53866 |
| **2** | 45926 |
| **3** | 25289 |
| **4** | 3069 |
| **5** | 7688 |
| **...** | ... |
| **4996** | 56831 |
| **4997** | 4625 |
| **4998** | 52007 |
| **4999** | 52595 |
| **5000** | 47848 |

5000 rows × 1 columns

```
answer_df.to_csv('submission_data_final.csv')
```

## E'tiboringiz uchun Tashakkur