1 importing library

```
import pandas as pd
```

```
import sklearn
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import learning_curve

from feature_engine.datetime import DatetimeFeatures

from xgboost import XGBRegressor

import joblib

import matplotlib.pyplot as plt
```

```
from xgboost import XGBRegressor
```

```
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
```

```
pd.set_option("display.max_columns", None)
```

```
sklearn.set_config(transform_output="default")
```

```
train_df =pd.read_csv("C://python program//ML_website//data//train.csv")
```

```
val_df =pd.read_csv("C://python program//ML_website//data//val.csv")
```

```
test_df =pd.read_csv("C://python program//ML_website//data//test.csv")
```

```
train_df
```

	airline	date_of_journey	source	destination	dep_time	arrival_time	duration	total_stops	additional_info	price
0	Jet Airways	2019-06-21	Mumbai	Hyderabad	10:20:00	11:50:00	90	0.0	In-flight meal not included	4995
1	Air India	2019-05-18	Delhi	Cochin	09:00:00	07:40:00	1360	1.0	No Info	8372
2	Air India	2019-06-12	Kolkata	Banglore	09:10:00	11:05:00	1555	2.0	No Info	6117
3	Vistara	2019-04-01	Kolkata	Banglore	20:20:00	22:55:00	1595	1.0	No Info	7770
4	Vistara	2019-06-06	Kolkata	Banglore	17:00:00	10:45:00	1065	1.0	No Info	9187
...	...	...	...	...	...	...	...	...	...	...
635	Air Asia	2019-04-12	Banglore	Delhi	04:55:00	07:45:00	170	0.0	No Info	4282
636	Jet Airways	2019-05-09	Kolkata	Banglore	09:35:00	21:05:00	690	1.0	No Info	13067
637	Indigo	2019-05-15	Banglore	Delhi	06:05:00	08:50:00	165	0.0	No Info	4423
638	Multiple Carriers	2019-05-15	Delhi	Cochin	08:45:00	21:00:00	735	1.0	No Info	7670
639	Jet Airways	2019-05-21	Kolkata	Banglore	20:00:00	12:00:00	960	1.0	In-flight meal not included	10844

640 rows × 10 columns

```
#split data
```

```
def split_data(data):
    X=data.drop(columns="price")
    y=data.price.copy()
    return (X,y)
```

```
X_train,y_train=split_data(train_df)
```

```
y_train
```

```
0       4995
1       8372
2       6117
3       7770
4       9187
       ...
635     4282
636    13067
637     4423
638     7670
639    10844
Name: price, Length: 640, dtype: int64
```

```
X_val,y_val=split_data(train_df)
```

```
X_test,y_test=split_data(train_df)
```

```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 640 entries, 0 to 639
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   airline          640 non-null    object
 1   date_of_journey  640 non-null    object
 2   source           640 non-null    object
 3   destination      640 non-null    object
 4   dep_time         640 non-null    object
 5   arrival_time     640 non-null    object
 6   duration         640 non-null    int64 
 7   total_stops      640 non-null    float64
 8   additional_info  640 non-null    object
dtypes: float64(1), int64(1), object(7)
memory usage: 45.1+ KB
```

```
# data_preprocessing
dt_col=["date_of_journey","dep_time","arrival_time"]
num_col=["duration","total_stops"]
cat_col=[col for col in X_train.columns if(col not in dt_col) and (col not in num_col)]
```

```
dt_col
```

```
['date_of_journey', 'dep_time', 'arrival_time']
```

```
num_col
```

```
['duration', 'total_stops']
```

```
num_transformer = Pipeline(steps=[
  ("imputer", SimpleImputer(strategy="median")),
  ("scaler", StandardScaler())
])

cat_transformer = Pipeline(steps=[
  ("imputer", SimpleImputer(strategy="most_frequent")),
  ("encoder", OneHotEncoder(sparse_output=False, handle_unknown="ignore"))
])

doj_transformer = Pipeline(steps=[
  ("imputer", SimpleImputer(strategy="most_frequent")),
  ("extractor", DatetimeFeatures(features_to_extract=["month", "week", "day_of_week", "day_of_month"], format="mixed")),
  ("scaler", StandardScaler())
```

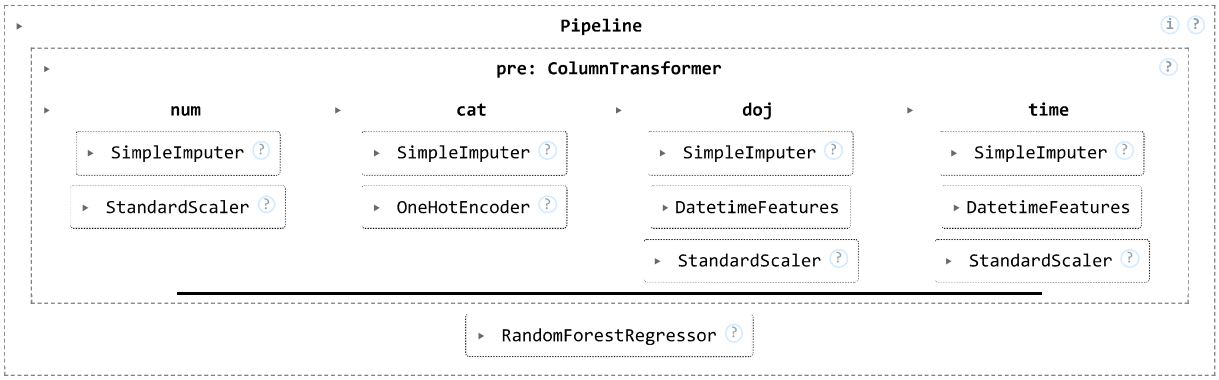
```
])

time_transformer = Pipeline(steps=[
  ("imputer", SimpleImputer(strategy="most_frequent")),
  ("extractor", DatetimeFeatures(features_to_extract=["hour", "minute"], format="mixed")),
  ("scaler", StandardScaler())
])


preprocessor = ColumnTransformer(transformers=[
  ("num", num_transformer, num_col),
  ("cat", cat_transformer, cat_col),
  ("doj", doj_transformer, ["date_of_journey"]),
  ("time", time_transformer, ["dep_time", "arrival_time"])
])


preprocessor.fit_transform(X_train)
```

```
array([[-1.09591823, -1.21213152,  0.        , ..., -0.14005709,
        -0.34523131,  1.49385907],
       [ 1.43569944,  0.31797533,  0.        , ..., -1.22986299,
        -0.93560684,  0.89104078],
       [ 1.82441239,  1.84808218,  0.        , ..., -0.68496004,
        -0.34523131, -1.21882323],
       ...,
       [-0.94641325, -1.21213152,  0.        , ..., -0.95741152,
        -0.78801296,  1.49385907],
       [ 0.18982461,  0.31797533,  0.        , ...,  1.22220029,
         1.1307075 , -1.52023237],
       [ 0.63833955,  0.31797533,  0.        , ..., -1.22986299,
        -0.19763743, -1.52023237]])
```

```
algorithms = {
    "Linear Regression": LinearRegression(),
    "Support Vector Machine": SVR(),
    "Random Forest": RandomForestRegressor(n_estimators=10),
    "XG Boost": XGBRegressor(n_estimators=10)
}


data = pd.concat([train_df, val_df], axis=0)

X_data, y_data = split_data(data)
print(X_data.shape, y_data.shape)
```

```
(800, 9) (800,)
```

```
# for name, alg in algorithms.items():
#   plot_learning_curves(name, alg)


# model training
model = Pipeline(steps=[
    ("pre", preprocessor),
    ("rf", RandomForestRegressor(n_estimators=10))
])


model.fit(X_data, y_data)
```



```
# model evauation
def evaluate_model(X, y):
    y_pred = model.predict(X)
    return r2_score(y, y_pred)
```

```
print(f"R2 score on Training data is = {evaluate_model(X_data, y_data)}")
```

```
R2 score on Training data is = 0.9397709258337793
```

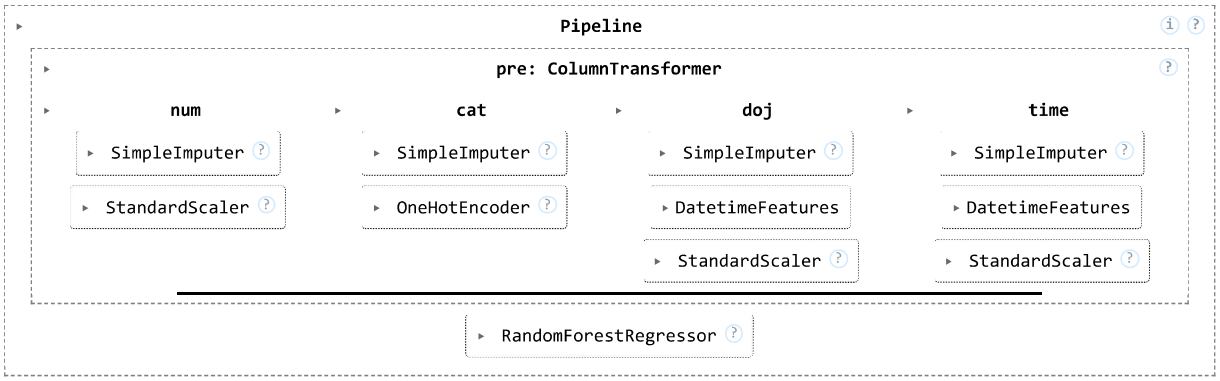
```
print(f"R2 score on Test data is = {evaluate_model(X_test, y_test)}")
```

```
R2 score on Test data is = 0.9318808940945664
```

```
# model persistance
joblib.dump(model, "model.joblib")
```

```
['model.joblib']
```

```
saved_model = joblib.load("model.joblib")
saved_model
```




```
y_pred = saved_model.predict(X_test)

r2_score(y_test, y_pred)
```

```
0.9318808940945664
```