ANS 1

In [1]:
```
import seaborn as sns
df=sns.load_dataset('tips')
df.head()
```

Out[1]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [2]:
```
df.isnull().sum()
```

Out[2]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

Here no missing values

In [3]:
```
X=df.drop(labels=['time'],axis=1)
y=df.time
```

In [4]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_st
```

In [5]:
```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [6]:
```
categorical_cols = ['sex', 'smoker','day']
numerical_cols = ['total_bill', 'tip','size']
```

In [7]:
```
num_pipeline=Pipeline(
    steps=[
        ('imputer',SimpleImputer(strategy='mean')),
        ('scaler',StandardScaler())
    ]

)
cat_pipeline=Pipeline(
                steps=[
                ('imputer',SimpleImputer(strategy='most_frequent')),
                ('onehotencoder',OneHotEncoder())
                ]
            )
```

In [8]:
```
preprocessor=ColumnTransformer([
    ('num_pipeline',num_pipeline,numerical_cols),
    ('cat_pipeline',cat_pipeline,categorical_cols)

])
```

In [9]:
```
X_train=preprocessor.fit_transform(X_train)
X_test=preprocessor.transform(X_test)
```

In [10]:
```
from sklearn.ensemble import RandomForestClassifier
models={
    'Random Forest':RandomForestClassifier(),
}

from sklearn.metrics import accuracy_score
```

In [11]:
```
def evaluate_model(X_train,y_train,X_test,y_test,models):

    report = {}
    for i in range(len(models)):
        model = list(models.values())[i]

        model.fit(X_train,y_train)

        y_test_pred =model.predict(X_test)

        test_model_score = accuracy_score(y_test,y_test_pred)

        report[list(models.keys())[i]] =  test_model_score

    return report
```

In [12]:
```
evaluate_model(X_train,y_train,X_test,y_test,models)
```

Out[12]:
```
{'Random Forest': 0.9591836734693877}
```

In [ ]:

In [ ]:

In [ ]:

ANS 2

In [13]:
```
import seaborn as sns
df=sns.load_dataset('iris')
df.head()
```

Out[13]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [14]:
```
x = df.drop('species',axis = 1)
y = df['species']
```

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_s
```

In [16]:
```
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
```

In [17]:
```
models={
    'Random Forest':RandomForestClassifier(),
    'Logistic Regression':LogisticRegression(),
   }
```

In [18]:
```
voting_classifier = VotingClassifier(
    estimators=[('rf', RandomForestClassifier()), ('lr',LogisticRegression
    voting='hard'
)

pipeline = Pipeline([
    ('voting', voting_classifier)
])
```

In [19]:
```
pipeline.fit(x_train, y_train)


y_pred = pipeline.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 1.0
```