```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from  sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

```
from google.colab import drive
drive.mount('/content/drive')
file_path = '/content/drive/MyDrive/heart.csv'
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
file_path = '/content/drive/MyDrive/heart.csv'
```

```
heart_test = pd.read_csv(file_path)
```

```
print(heart_test.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
None
```

```
heart_test_list= list(heart_test.columns)
print(heart_test_list)
```

```
['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh', 'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output']
```

```
heart_test.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

Next steps: Generate code with heart_test View recommended plots

```
heart_test.describe()
```

	age	sex	cp	trtbps	chol	fbs	restec
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.52805
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.52586
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.00000

```
print(heart_test.isnull().sum())
```

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

```
features=list(set(heart_test.columns)-set(['output']))
```

```
print(features)
```

```
['chol', 'oldpeak', 'fbs', 'trtbps', 'exng', 'thall', 'thalachh', 'cp', 'restecg', 'age', 'sex', 'slp', 'caa']
```

```
target=(['output'])
```

```
print(target)
```

```
['output']
```

```
y=heart_test[target].values
```

```
X=heart_test[features].values
```

```
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.3,random_state=0)
```

```
scaler=StandardScaler()
```

```
scaler.fit(train_X)
```

```
StandardScaler
StandardScaler()
```

```
train_X=scaler.transform(train_X)
```

```
test_X=scaler.transform(test_X)
```

LOGISTIC REGRESSION OF THE SAMPLE

```
Log_model=LogisticRegression()
```

```
Log_model.fit(train_X,train_y.ravel())
```

```
LogisticRegression
LogisticRegression()
```

```
y_log_pred=Log_model.predict(test_X)
```

```
CM_log=confusion_matrix(y_log_pred,test_y)
```

```
print(CM_log)
```

```
[[32  5]
 [12 42]]
```

```
from sklearn.metrics import accuracy_score
accuracy_log=accuracy_score(y_log_pred,test_y)
print("Accuracy score of the prediction by Logistic Regression Algorithm is", accuracy_log ,
      '.\n logistic regression model of the sample is', accuracy_log*100, '% accurate.')
```

```
Accuracy score of the prediction by Logistic Regression Algorithm is 0.8131868131868132 .
 logistic regression model of the sample is 81.31868131868131 % accurate.
```

K-NEAREST NEIGHBORS CLASSIFIER

```
KNN = KNeighborsClassifier(n_neighbors=2)
```

```
KNN.fit(train_X,train_y.ravel())
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=2)
```

```
KNN_pred=KNN.predict(test_X)
```

```
confusion_matrix=confusion_matrix(test_y,KNN_pred)
```

```
print(confusion_matrix)
```

```
[[39  5]
 [14 33]]
```

```
accuracy_score_KNN=(39+33)/(39+5+14+33)
print("Accuracy score of the prediction by K-Nearest Neighbors Classifier Algorithm is", accuracy_score_KNN ,'.\n K-Nearest Neighbors C
```

```
Accuracy score of the prediction by K-Nearest Neighbors Classifier Algorithm is 0.7912087912087912 .
 K-Nearest Neighbors Classifier model of the sample is 79.12087912087912 % accurate.
```

SUPPORT VECTOR MACHINE

```
clf=SVC(kernel='linear')
```

```
clf.fit(train_X,train_y.ravel())
```

```
SVC
SVC(kernel='linear')
```

```
SVM_pred=clf.predict(test_X)
```

```
from sklearn.metrics import accuracy_score
accuracy_SVM=accuracy_score(test_y, SVM_pred)
```

```
print("Accuracy score of the prediction by Support Vector Machine Algorithm is", accuracy_SVM ,'.\n Support Vector Machine model of the
```

```
Accuracy score of the prediction by Support Vector Machine Algorithm is 0.8021978021978022 .
 Support Vector Machine model of the sample is 80.21978021978022 % accurate.
```

DECISION TREE CLASSIFIER

```
dt_classifier = DecisionTreeClassifier(criterion='gini')
```

```
dt_classifier.fit(train_X, train_y)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pred_dt = dt_classifier.predict(test_X)
```

```
from sklearn.metrics import accuracy_score
accuracy_dt = accuracy_score(test_y, pred_dt)
print("Accuracy score of the prediction by Decision Tree Classifier Algorithm is", accuracy_dt ,'.\n Decision Tree Classifier model of t
```

```
Accuracy score of the prediction by Decision Tree Classifier Algorithm is 0.7142857142857143 .
 Decision Tree Classifier model of the sample is 71.42857142857143 % accurate.
```

RANDOM FOREST CLASSIFIER

```
Classifier= RandomForestClassifier(random_state=90)
```

```
Classifier.fit(train_X,train_y.ravel())
```

```
RandomForestClassifier
RandomForestClassifier(random_state=90)
```

```
params= {'max_depth':[15,20,25],
       'max_features':['auto','sqrt'],
       'min_samples_split':[15,20,25],
       'min_samples_leaf':[5,10],
       'n_estimators':[10,25,30]}
```

```
grid_search_result=GridSearchCV(estimator=Classifier,param_grid=params, cv=5, scoring='accuracy')
```

```
grid_search_result.fit(train_X, train_y.ravel())
```

Show hidden output

```
best_model=grid_search_result.best_estimator_
print(best_model)
```

```
RandomForestClassifier(max_depth=15, max_features='auto', min_samples_leaf=10,
                       min_samples_split=15, n_estimators=30, random_state=90)
```

```
accuracy_rf=grid_search_result.best_score_
print("Accuracy score of the prediction by Random Forest Classifier Algorithm is", accuracy_rf ,'.\n Random Forest Classifier model of th
```

```
Accuracy score of the prediction by Random Forest Classifier Algorithm is 0.8446290143964562 .
 Random Forest Classifier model of the sample is 84.46290143964562 % accurate.
```

```
accuracies = {
    'Logistic Regression': accuracy_log,
    'K-Nearest Neighbours': accuracy_score_KNN,
    'Support Vector Machine': accuracy_SVM,
    'Decision Tree': accuracy_dt,
    'Random Forest': accuracy_rf
}
```

```
best_model = max(accuracies)
for key, value in accuracies.items():
    print(f'{key}: {value}')
print('\nBest model that can be used for prediction of heart disease of patients with highest accuracy score is -', best_model)
```

```
Logistic Regression: 0.8131868131868132

K-Nearest Neighbours: 0.7912087912087912

Support Vector Machine: 0.8021978021978022

Decision Tree: 0.7142857142857143

Random Forest: 0.8446290143964562

Best model that can be used for prediction of heart disease of patients with highest accuracy score is - Logistic Regression
```

Start coding or generate with AI.