```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np # Added for numpy operations
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression # Added for Logistic Regression
from sklearn.neighbors import KNeighborsClassifier # Added for KNN Classifier
from sklearn.metrics import confusion_matrix # Added for confusion_matrix

accuracies = {} # Initialized accuracies dictionary
```

```
df = pd.read_csv('/content/heart.csv')
```

```
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Next steps: Generate code with df New interactive sheet

```
df.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   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
df.shape
```

```
(303, 14)
```

```
df.isnull().sum()
```

```
          0
age       0
sex       0
cp        0
trestbps  0
chol      0
fbs       0
restecg   0
thalach   0
exang     0
oldpeak   0
slope     0
ca        0
thal      0
target    0

dtype: int64
```

```
df.duplicated().sum()
```

```
np.int64(1)
```

```
df.drop_duplicates(inplace=True, keep = 'first')
```

```
df.shape
```

```
(302, 14)
```

```
df.describe().T
```

```
          count  mean        std        min    25%     50%    75%     max
age       302.0  54.420530   9.047970   29.0   48.00   55.5   61.00   77.0
sex       302.0  0.682119    0.466426   0.0    0.00    1.0    1.00    1.0
cp        302.0  0.963576    1.032044   0.0    0.00    1.0    2.00    3.0
trestbps  302.0  131.602649  17.563394  94.0   120.00  130.0  140.00  200.0
chol      302.0  246.500000  51.753489  126.0  211.00  240.5  274.75  564.0
fbs       302.0  0.149007    0.356686   0.0    0.00    0.0    0.00    1.0
restecg   302.0  0.526490    0.526027   0.0    0.00    1.0    1.00    2.0
thalach   302.0  149.569536  22.903527  71.0   133.25  152.5  166.00  202.0
exang     302.0  0.327815    0.470196   0.0    0.00    0.0    1.00    1.0
oldpeak   302.0  1.043046    1.161452   0.0    0.00    0.8    1.60    6.2
slope     302.0  1.397351    0.616274   0.0    1.00    1.0    2.00    2.0
ca        302.0  0.718543    1.006748   0.0    0.00    0.0    1.00    4.0
thal      302.0  2.314570    0.613026   0.0    2.00    2.0    3.00    3.0
target    302.0  0.543046    0.498970   0.0    0.00    1.0    1.00    1.0
```

```
df['target'].value_counts()
```

```
        count
target
1       164
0       138

dtype: int64
```

```
df = df[df['ca'] < 4]  #CA IN KAGGLE 0<3 delete any num bigger than 3
DF = df[df['thal'] > 0] #THAL IN KAGGLE delete any num less than 0
print(df.shape)
print(f'len of data: {len(df)}')
```

```
(298, 14)
len of data: 298
```

```
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
df = df.rename(
    columns = {
        'cp' : 'chest_pain_type',
        'trestbps':'resting_blood_pressure',
               'chol': 'cholesterol',
               'fbs': 'fasting_blood_sugar',
               'restecg' : 'resting_electrocardiogram',
               'thalach': 'max_heart_rate_achieved',
               'exang': 'exercise_induced_angina',
               'oldpeak': 'st_depression',
               'slope': 'st_slope',
               'ca':'num_major_vessels',
               'thal': 'thalassemia'},
    errors = "raise")
```

```
df.head()
```

	age	sex	chest_pain_type	resting_blood_pressure	cholesterol	fasting_blood_sugar	resting_electrocardiogram	max_heart
0	63	1	3	145	233	1	0	
1	37	1	2	130	250	0	1	
2	41	0	1	130	204	0	0	
3	56	1	1	120	236	0	1	
4	57	0	0	120	354	0	1	

Next steps: Generate code with df New interactive sheet

```
df['sex'][df['sex'] == 0] = 'female'
df['sex'][df['sex'] == 1] = 'male'
```

```
/tmp/ipython-input-3284524103.py:1: FutureWarning: ChainedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases, but when using Copy-on-Write (whic
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single step and ensure this keeps updating

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view

  df['sex'][df['sex'] == 0] = 'female'
/tmp/ipython-input-3284524103.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  df['sex'][df['sex'] == 0] = 'female'
/tmp/ipython-input-3284524103.py:1: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an err
  df['sex'][df['sex'] == 0] = 'female'
```

```
df.dtypes
```

```
                                   0
age                            int64
sex                           object
chest_pain_type                int64
resting_blood_pressure         int64
cholesterol                    int64
fasting_blood_sugar            int64
resting_electrocardiogram      int64
max_heart_rate_achieved        int64
exercise_induced_angina        int64
st_depression                float64
st_slope                       int64
num_major_vessels              int64
thalassemia                    int64
target                         int64

dtype: object
```

```
count = 0
for i in df.dtypes:
    if i == 'object':
        count += 1

print(count)
```

```
1
```

```
df.describe(include='object')
```

```
         sex
count    298
unique     2
top     male
freq     202
```

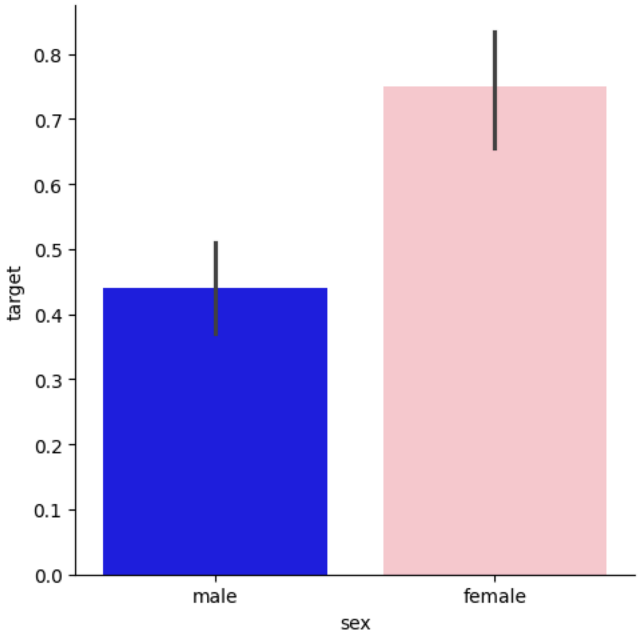
```
Start coding or generate with AI.
```

```
get_object_columns = [col for col in df.columns if df[col].dtypes == 'object']
get_object_columns
```

```
['sex']
```

```
plt.figure(figsize = (16,10))
for i in get_object_columns:
    sns.catplot(y = "target", x = i, hue = "sex",data = df, kind = "bar",
               palette={"male": "blue", "female": "pink"})
```

```
<Figure size 1600x1000 with 0 Axes>
```



```
def label_encode_cat_features(data, cat_features):
  label_encoder = LabelEncoder()
  data_encoded = data.copy()

  for col in cat_features:
    data_encoded[col] = label_encoder.fit_transform(data[col])
  data = data_encoded
  return data
```

```
cat_features = ['sex', 'fasting_blood_sugar', 'exercise_induced_angina', 'target',
                'chest_pain_type', 'resting_electrocardiogram', 'st_slope', 'thalassemia']
```

```
df = label_encode_cat_features(df, cat_features)
```

```
df.head()
```

	age	sex	chest_pain_type	resting_blood_pressure	cholesterol	fasting_blood_sugar	resting_electrocardiogram	max_heart
0	63	1	3	145	233	1	0	
1	37	1	2	130	250	0	1	
2	41	0	1	130	204	0	0	
3	56	1	1	120	236	0	1	
4	57	0	0	120	354	0	1	

Next steps: Generate code with df New interactive sheet

```
X = df.iloc[ : , :-1]
y = df.iloc[ : , -1]
```

```
X.head(2)
```

	age	sex	chest_pain_type	resting_blood_pressure	cholesterol	fasting_blood_sugar	resting_electrocardiogram	max_heart
0	63	1	3	145	233	1	0	
1	37	1	2	130	250	0	1	

Next steps: Generate code with X New interactive sheet

```
y.head(2)
```

```
    target
0        1
1        1

dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42, shuffle = True)
```

```
lr = LogisticRegression()
lr.fit(X_train,y_train)
acc = lr.score(X_test,y_test)*100

accuracies['Logistic Regression'] = acc
print("Test Accuracy {:.2f}%".format(acc))
```

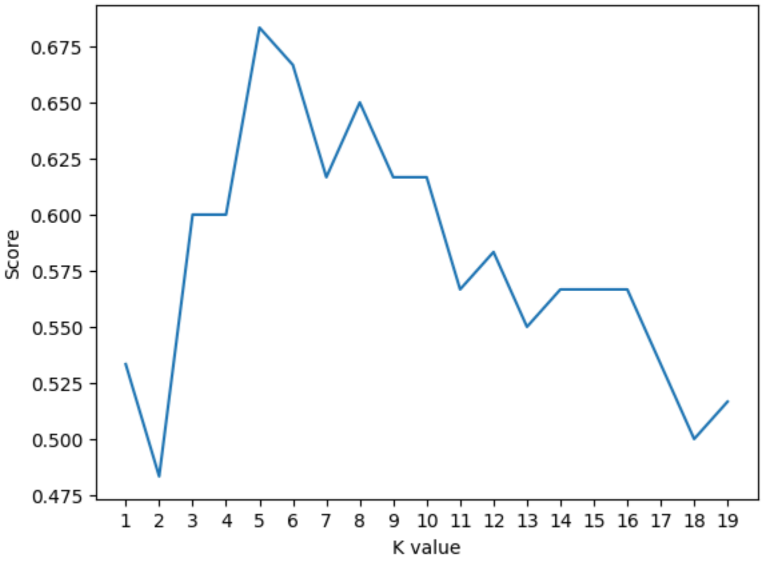
```
Test Accuracy 81.67%
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
scoreList = []
for i in range(1,20):
    knn = KNeighborsClassifier(n_neighbors = i)  # n_neighbors means k
    knn.fit(X_train, y_train)
    scoreList.append(knn.score(X_test, y_test))

plt.plot(range(1,20), scoreList)
plt.xticks(np.arange(1,20,1))
plt.xlabel("K value")
plt.ylabel("Score")
plt.show()

acc = max(scoreList)*100
accuracies['KNN'] = acc
print("Maximum KNN Score is {:.2f}%".format(acc))
```



```
Maximum KNN Score is 68.33%
```

```
accuracies
```

```
{'Logistic Regression': 81.66666666666667, 'KNN': 68.33333333333333}
```

```
from sklearn.svm import SVC

svc = SVC()
svc.fit(X_train, y_train)

acc = svc.score(X_test,y_test)*100
accuracies['SVC'] = acc
print(f"Test Accuracy of SVC Algorithm: {acc:.2f}%")
```

```
Test Accuracy of SVC Algorithm: 65.00%
```

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)

acc = nb.score(X_test,y_test)*100
accuracies['Naive Bayes'] = acc
print(f"Accuracy of Naive Bayes: {acc:.2f}%")
```

```
Accuracy of Naive Bayes: 83.33%
```

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

acc = dtc.score(X_test, y_test)*100
accuracies['Decision Tree'] = acc
print(f"Decision Tree Test Accuracy {acc:.2f}%")
```

```
Decision Tree Test Accuracy 70.00%
```

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 1000, random_state = 1)
rf.fit(X_train, y_train)

acc = rf.score(X_test,y_test)*100
accuracies['Random Forest'] = acc
print(f"Random Forest Algorithm Accuracy Score : {acc:.2f}%")
```

```
Random Forest Algorithm Accuracy Score : 85.00%
```

```
colors = ["skyblue", "salmon", "lightgreen", "gold", "lightcoral", "plum"]

sns.set_style("whitegrid")
plt.figure(figsize=(16, 5))
plt.yticks(np.arange(0, 110, 10))
plt.ylabel("Accuracy %")
plt.xlabel("Algorithms")


ax = sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors)


for i, v in enumerate(accuracies.values()):
    ax.text(i, v + 1, f"{v:.2f}%", color='black', ha='center', fontweight='bold')

plt.title("Model Accuracy Comparison")
plt.ylim(0, 110)
plt.show()
```

```
/tmp/ipython-input-2054633348.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  ax = sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors)
```

Model Accuracy Comparison

```
y_pred_lr  = lr.predict(X_test)
y_pred_knn = knn.predict(X_test)
y_pred_svc = svc.predict(X_test)
y_pred_nb  = nb.predict(X_test)
y_pred_dtc = dtc.predict(X_test)
y_pred_rf  = rf.predict(X_test)
```

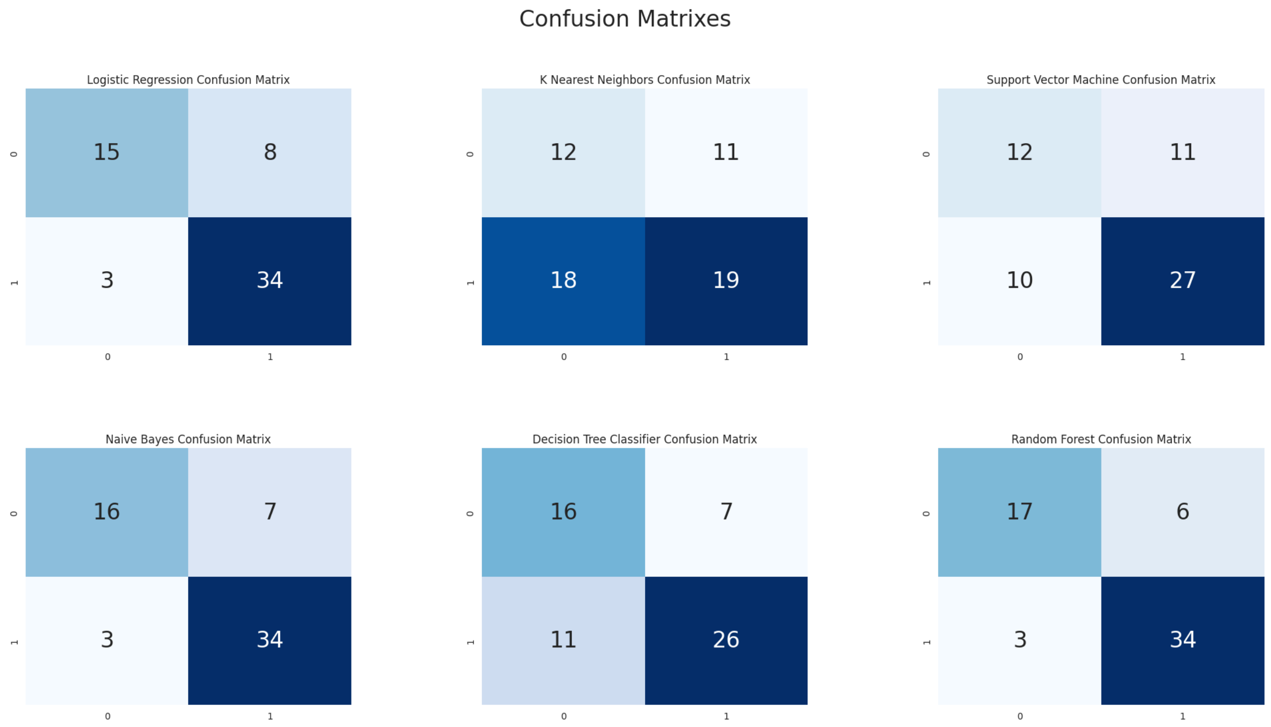
```
cm_lr = confusion_matrix(y_test,y_pred_lr)
cm_knn = confusion_matrix(y_test,y_pred_knn)
cm_svc = confusion_matrix(y_test,y_pred_svc)
cm_nb = confusion_matrix(y_test,y_pred_nb)
cm_dtc = confusion_matrix(y_test,y_pred_dtc)
cm_rf = confusion_matrix(y_test,y_pred_rf)
```

```
models = [
    ("Logistic Regression", cm_lr),
    ("K Nearest Neighbors", cm_knn),
    ("Support Vector Machine", cm_svc),
    ("Naive Bayes", cm_nb),
    ("Decision Tree Classifier", cm_dtc),
    ("Random Forest", cm_rf)
]

plt.figure(figsize=(24, 12))
plt.suptitle("Confusion Matrixes", fontsize=24)
plt.subplots_adjust(wspace=0.4, hspace=0.4)

for i, (title, cm) in enumerate(models, 1):
    plt.subplot(2, 3, i)
    plt.title(f"{title} Confusion Matrix")
    sns.heatmap(cm, annot=True, cmap="Blues", fmt="d", cbar=False, annot_kws={"size": 24})

plt.show()
```


Confusion Matrixes
Logistic Regression Confusion Matrix
K Nearest Neighbors Confusion Matrix
Support Vector Machine Confusion Matrix
Naive Bayes Confusion Matrix
Decision Tree Classifier Confusion Matrix
Random Forest Confusion Matrix