FINAL LAB - CAC 3

SUBMITTED BY:

Name : Kavitha.S
Reg no : 21122033
Class : 2MSDS

PROBLEM STATEMENT:

• Find out a Dataset, and compare at least two different algorithms and choose the best one
• Use suitable Data Preprocessing and Feature Selection/Engineering Methods
• Fine tune the model and hyper parameters and Finalise the Model
• Make the model deployment-ready by giving User-Input provision

PROBLEM DEFINITION AND APPROACH:

Import an external dataset. Perform various EDA to learn more about the dataset. Based on the correlation between the various the dependent and independent variables, and other deciding factors, we did feature engineering. In order for the data to range between a certain value, we did scaling. We performed three machine learning algorithm on the dataset. Based on the accuracy score, we decided the best model. And also did hyperparameter tuning inorder to increase the accuracy of the model created. We added user-input provision and made the model ready for deployment.

In [1]:
```
pip install plotly==5.8.0
```

```
Requirement already satisfied: plotly==5.8.0 in c:\users\sridhar\anaconda3\lib
\site-packages (5.8.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\sridhar\anaconda3\li
b\site-packages (from plotly==5.8.0) (8.0.1)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, roc_curve
from sklearn.model_selection import cross_val_score
```

Heart Attack Analysis and Prediction

This dataset contains information about people and there chances of having a heart stroke. Based on the various symptoms or other biological data of an individual, we predict if they will get heart attack or not.

Data Dictionary

age - Age of the patient
sex - Sex of the patient , Male = 1 and female =0
cp - Chest pain type ~ 0 = Typical Angina, 1 = Atypical Angina, 2 = Non-anginal Pain, 3 = Asymptomatic
trtbps - Resting blood pressure (in mm Hg)
chol - Cholestoral in mg/dl fetched via BMI sensor
fbs - (fasting blood sugar > 120 mg/dl) ~ 1 = True, 0 = False
restecg - Resting electrocardiographic results ~ 0 = Normal, 1 = ST-T wave normality, 2 = Left ventricular hypertrophy
thalachh - Maximum heart rate achieved
oldpeak - Previous peak
slp - Slope
caa - Number of major vessels
thall - Thalium Stress Test result ~ (0,3)
exng - Exercise induced angina ~ 1 = Yes, 0 = No
output - Target variable , 1 - Got heart attack, 0 - Did not

In [6]:
```
heart = pd.read_csv("heart.csv")
```

In [7]:
```
1 heart.head()
```

Out[7]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
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

Exploratory Data Analysis

In [8]:
```
1 heart.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
sex	303.0	0.683168	0.466011	0.0	0.0	1.0	1.0	1.0
cp	303.0	0.966997	1.032052	0.0	0.0	1.0	2.0	3.0
trtbps	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
fbs	303.0	0.148515	0.356198	0.0	0.0	0.0	0.0	1.0
restecg	303.0	0.528053	0.525860	0.0	0.0	1.0	1.0	2.0
thalachh	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
exng	303.0	0.326733	0.469794	0.0	0.0	0.0	1.0	1.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2
slp	303.0	1.399340	0.616226	0.0	1.0	1.0	2.0	2.0
caa	303.0	0.729373	1.022606	0.0	0.0	0.0	1.0	4.0
thall	303.0	2.313531	0.612277	0.0	2.0	2.0	3.0	3.0
output	303.0	0.544554	0.498835	0.0	0.0	1.0	1.0	1.0

Observation:

The average blood pressure of an individual is 130 whereas the maximun value goes upto 200.
The average heart rate of the group is 152, whereas overall it ranges between 133 to 202
Age of the group varies from 29 to 77 and the mean age is 55.5

In [9]:
```
print("The shape of the dataset is : ", heart.shape)
```

```
The shape of the dataset is :  (303, 14)
```

In [10]:
```
dict = {}
for i in list(heart.columns):
    dict[i] = heart[i].value_counts().shape[0]

pd.DataFrame(dict,index=["unique count"]).transpose()
```

Out[10]:

	unique count
age	41
sex	2
cp	4
trtbps	49
chol	152
fbs	2
restecg	3
thalachh	91
exng	2
oldpeak	40
slp	3
caa	5
thall	4
output	2

We check for duplicated rows in the dataset. And delete the duplicate.

In [11]:
```
heart.duplicated().sum()
```

Out[11]: 1

In [12]:
```
heart.drop_duplicates(inplace=True)
print("The shape of the dataset is : ", heart.shape)
```

```
The shape of the dataset is :  (302, 14)
```

In [13]:
```
heart.isnull().sum()
```

Out[13]:
```
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
```

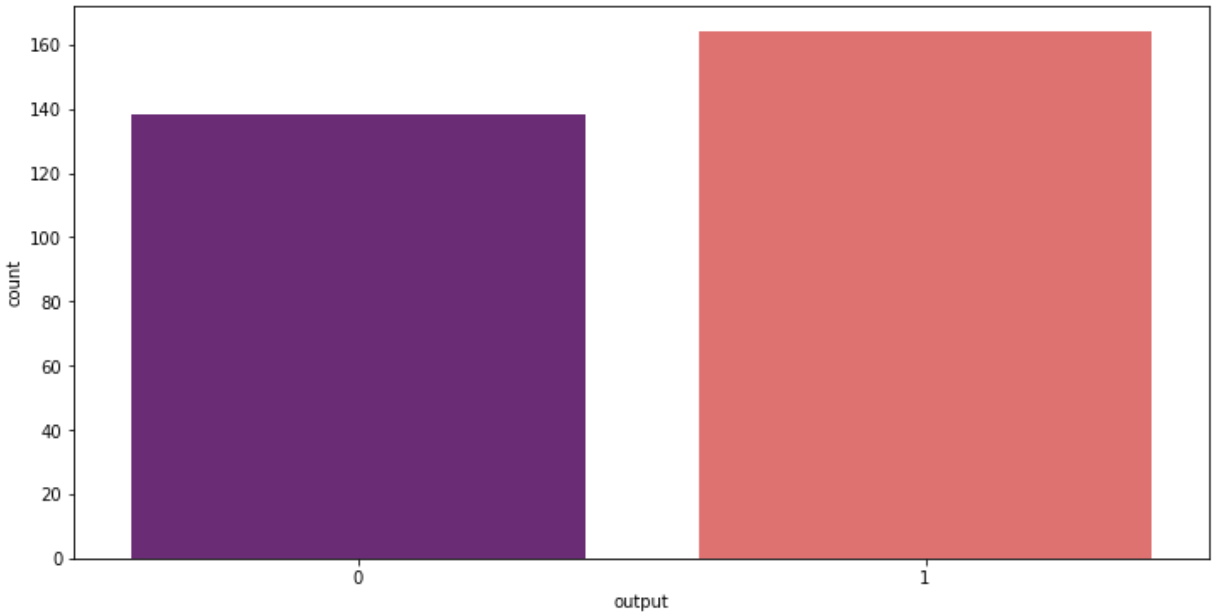
There is no null value in the dataset

We have both categorical and Continuous variables in the dataset. So we are seperating it.

In [14]:
```
cat_cols = ['sex','exng','caa','cp','fbs','restecg','slp','thall']
con_cols = ['age','trtbps','chol','thalachh','oldpeak']
target_col = ["output"]
print("The categorial cols are : ", cat_cols)
print("The continuous cols are : ", con_cols)
print("The target variable is :  ", target_col)
```

```
The categorial cols are :  ['sex', 'exng', 'caa', 'cp', 'fbs', 'restecg', 'sl
p', 'thall']
The continuous cols are :  ['age', 'trtbps', 'chol', 'thalachh', 'oldpeak']
The target variable is :   ['output']
```

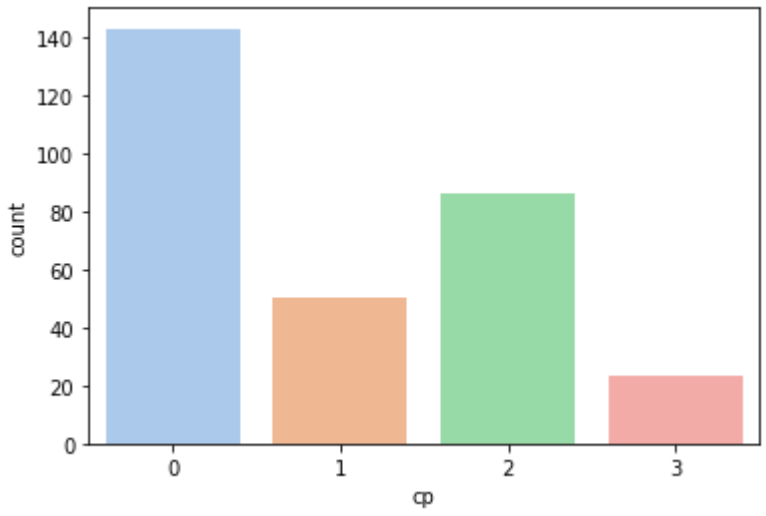
In [15]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x="output", data=heart, palette='magma');
```



As you can see, the above graph is the countplot of the target variable. The people who got the heart attack are more. There's a slight chance that the model will be more trained on people who have a higher chance of getting heart attack.

In [16]:
```
ax=plt.axis()
sns.countplot(x='cp', data=heart, palette='pastel')
```

Out[16]:
```
<AxesSubplot:xlabel='cp', ylabel='count'>
```

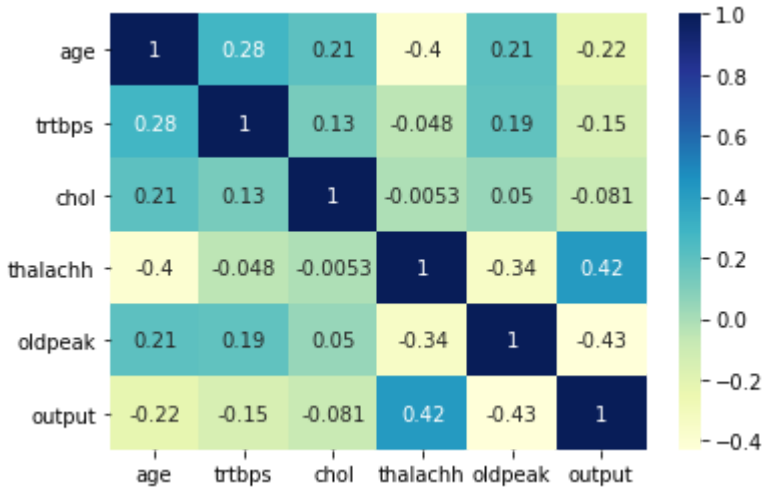


Above is the countplot of the four types of chest pain people had. As you can see People had Typical Angina more.

In [18]:
```
df = ['age','trtbps','chol','thalachh','oldpeak','output']
```

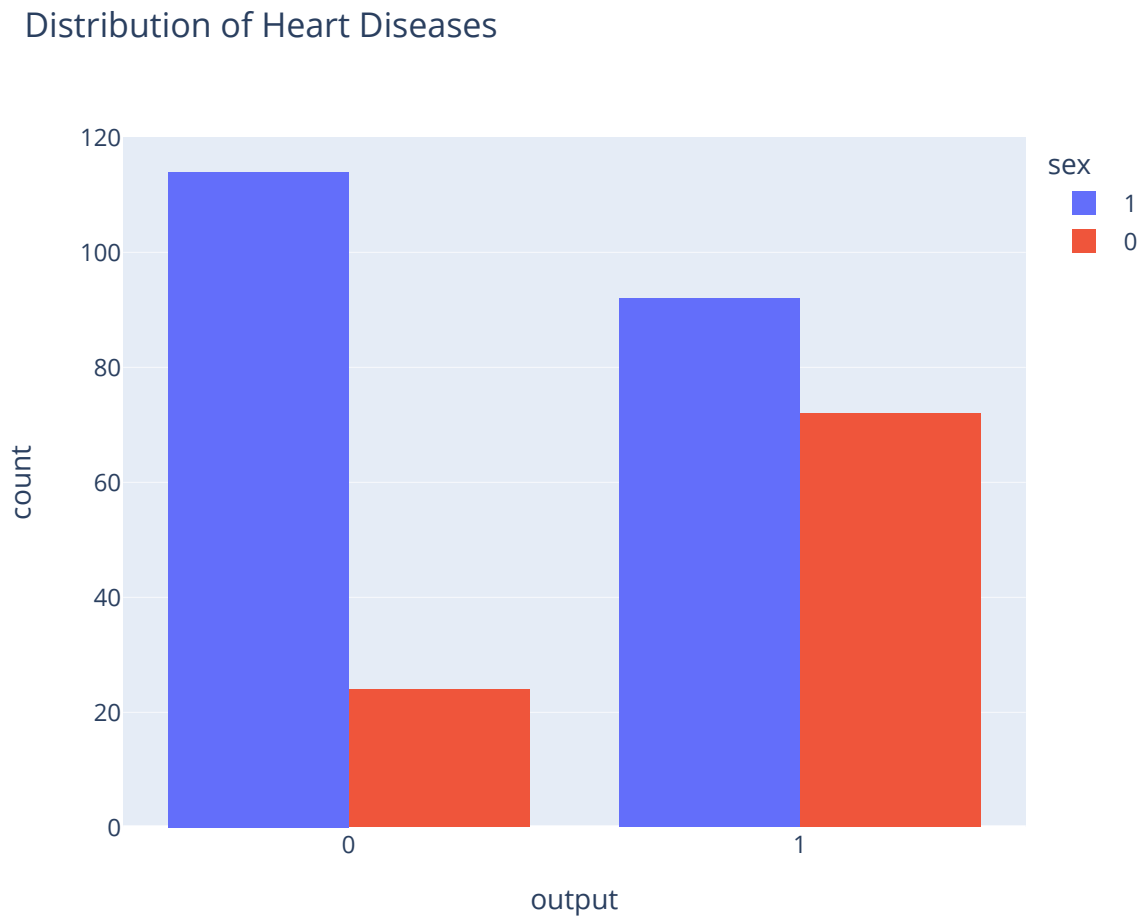
In [19]:
```
heart_corr = heart[df].corr().transpose()
sns.heatmap(heart_corr,annot=True,cmap='YlGnBu')
```

Out[19]: <AxesSubplot:>



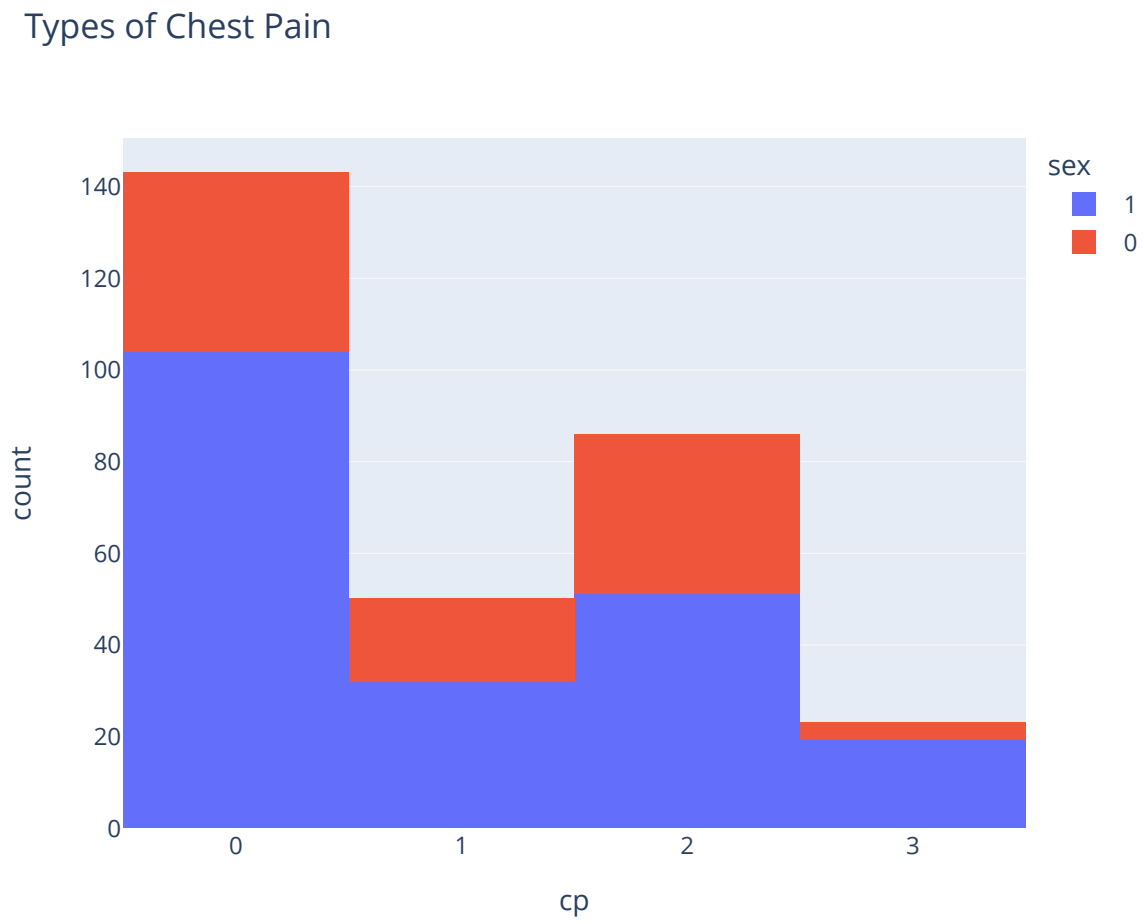
Above is the correlation plot between the continuous variables and the dependent variable. The highest positive correlation is between the output and thalachh(Max heart rate achieved). That is the higher the heart rate, the more the chance of getting a heart attack.
The highest negative correlation is the oldpeak(Previous peak). There is no apparent linear correlation between continuous variable according to the heatmap

In [20]:
```
fig=px.histogram(heart,
                 x="output",
                 color="sex",
                 hover_data=heart.columns,
                 title="Distribution of Heart Diseases",
                 barmode="group")
fig.show()
```



Above is the distribution of the heart disease based on the gender of the individual. Where we can see that male has the most number of heart attack.

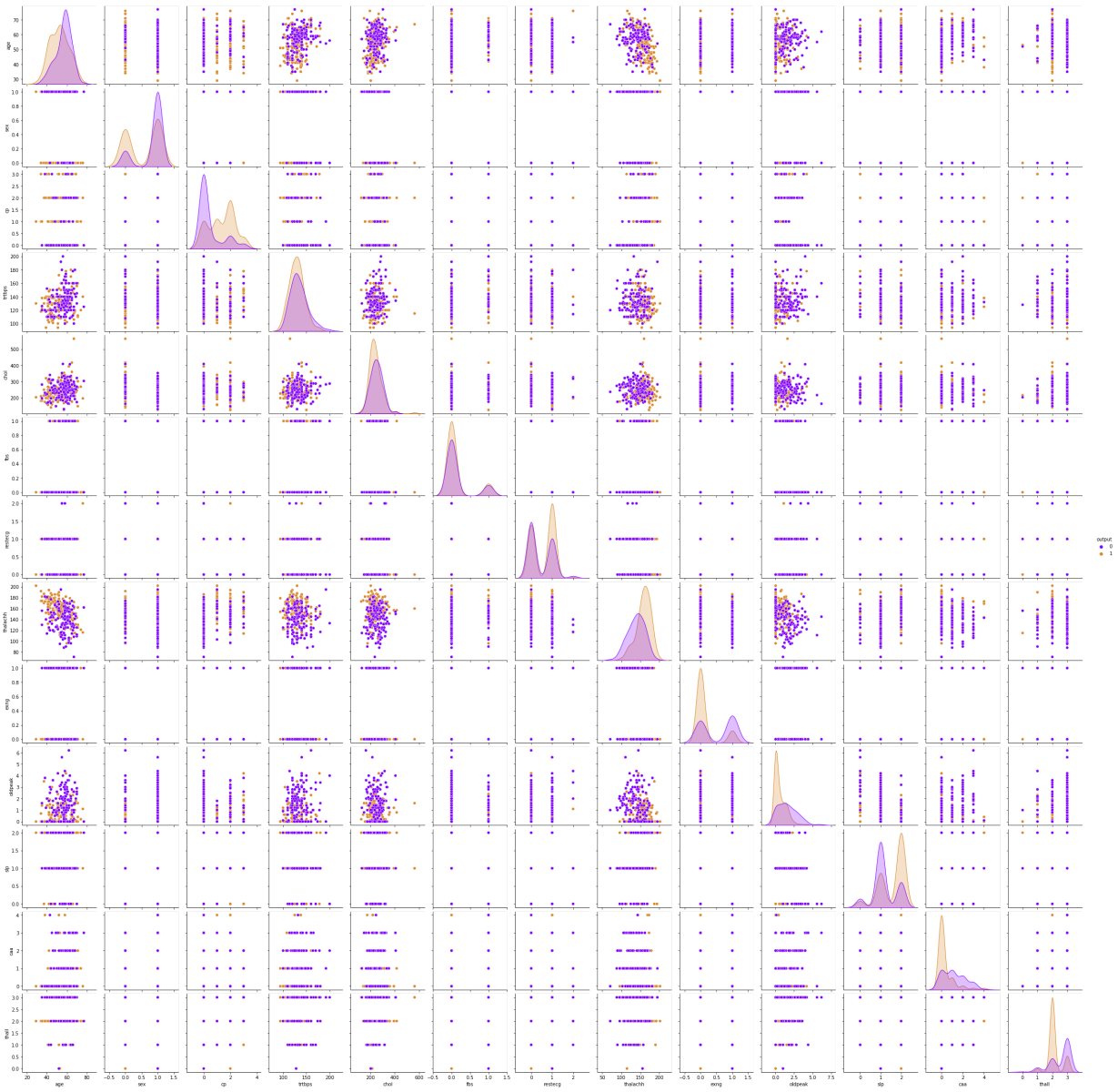
In [21]:
```
fig=px.histogram(heart, x="cp", color="sex", hover_data=heart.columns, title
fig.show()
```



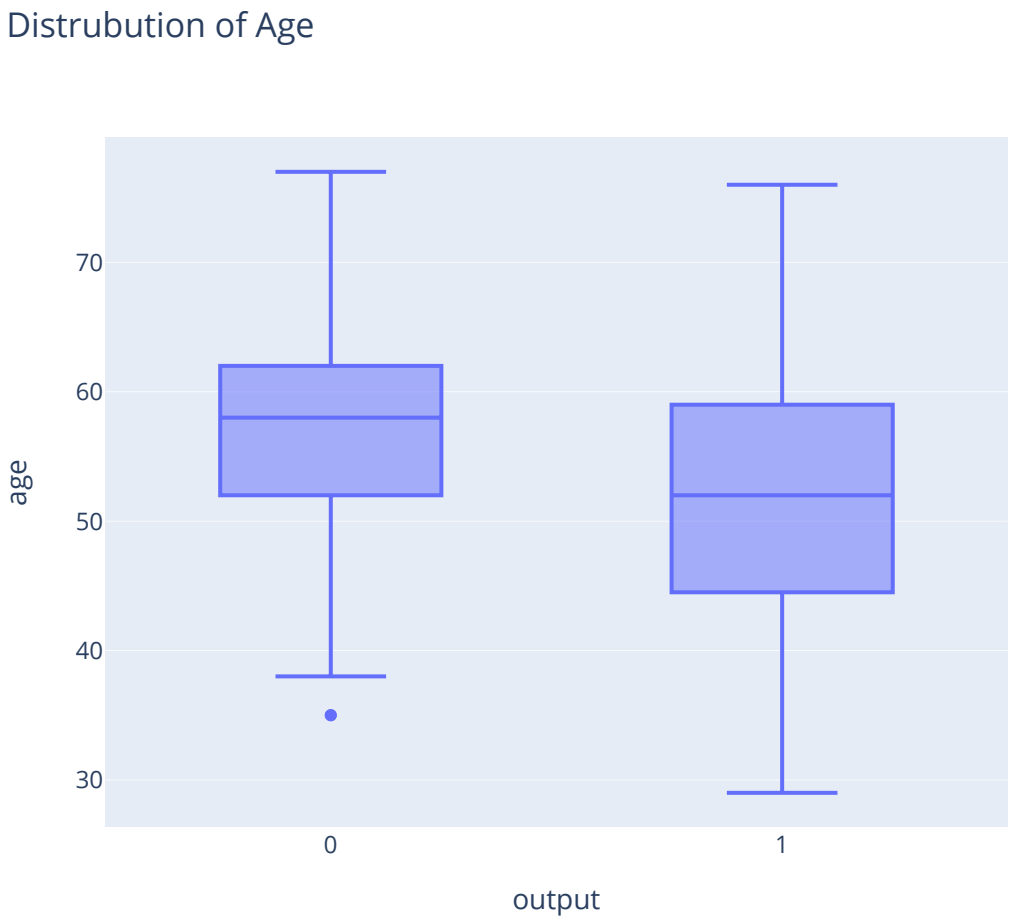
Above is the distribution of the various types of chest pains by the gender.

Pairplot according to target variable - one plot to rule them all

In [23]:
```
sns.pairplot(heart,hue='output',palette = ["#8000ff","#da8829"])
plt.show()
```



In [22]:
```
fig = px.box(heart,y="age",x="output",title=f"Distrubution of Age")
fig.show()
```



It is intuitive that elder people might have higher chances of heart attack but according to the distribution plot of age with respect to output, it is evident that this isn't the case.

Scaling the features

In [24]:
```
heart_copy = heart
```

In [25]:
```
X = heart_copy.drop(['output'],axis=1)
y = heart_copy[['output']]
```

In [26]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X[con_cols] = scaler.fit_transform(X[con_cols])
X.head()
```

Out[26]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	tha
0	0.949794	1	3	0.764066	-0.261285	1	0	0.018826	0	1.084022	0	0	
1	-1.928548	1	2	-0.091401	0.067741	0	1	1.636979	0	2.118926	0	0	
2	-1.485726	0	1	-0.091401	-0.822564	0	0	0.980971	0	0.307844	2	0	
3	0.174856	1	1	-0.661712	-0.203222	0	1	1.243374	0	-0.209608	2	0	
4	0.285561	0	0	-0.661712	2.080602	0	1	0.587366	1	-0.382092	2	0	

Performing Train Test split

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, ra
print("The shape of X_train is      ", X_train.shape)
print("The shape of X_test is       ",X_test.shape)
print("The shape of y_train is      ",y_train.shape)
print("The shape of y_test is       ",y_test.shape)
```

```
The shape of X_train is       (241, 13)
The shape of X_test is        (61, 13)
The shape of y_train is       (241, 1)
The shape of y_test is        (61, 1)
```

```
Creating an empty list to store the accuracy score.
```

In [29]:
```
model = ['SVC','LogisticRegression','DecisionTreeClassifier']
predicted =[]
```

In [30]:
```
clf = SVC(kernel='linear', C=1, random_state=42).fit(X_train,y_train)

# predicting the values
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
# printing the test accuracy
print("The test accuracy score of SVM is ", acc)
predicted = np.append(predicted, acc)
```

```
The test accuracy score of SVM is  0.8524590163934426

C:\Users\SRIDHAR\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: Da
taConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().
```

In [31]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        29
           1       0.87      0.84      0.86        32

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61
```

In [32]:
```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
acc= accuracy_score(y_test, y_pred)
print("The test accuracy score of Logistic Regression is ",acc)
predicted = np.append(predicted, acc)
```

```
The test accuracy score of Logistic Regression is  0.8360655737704918

C:\Users\SRIDHAR\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: Da
taConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().
```

In [33]:
```
print(classification_report(y_test, y_pred))
```

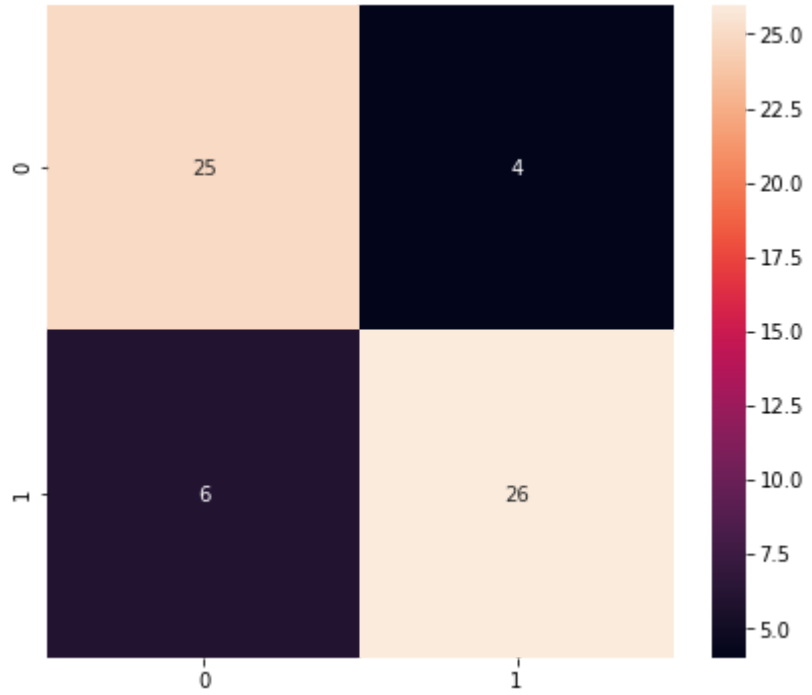
```
              precision    recall  f1-score   support

           0       0.81      0.86      0.83        29
           1       0.87      0.81      0.84        32

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61
```

In [34]:
```
from sklearn.metrics import confusion_matrix
conf = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(7,6))
sns.heatmap(conf,annot=True,fmt='d')
```

Out[34]: <AxesSubplot:>



In [35]:
```
dt = DecisionTreeClassifier(random_state = 42)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
acc =accuracy_score(y_test, y_pred)
print("The test accuracy score of Decision Tree is ", acc)
predicted = np.append(predicted, acc)
```

```
The test accuracy score of Decision Tree is  0.7704918032786885
```

In [36]:
```
print(classification_report(y_test, y_pred))
```

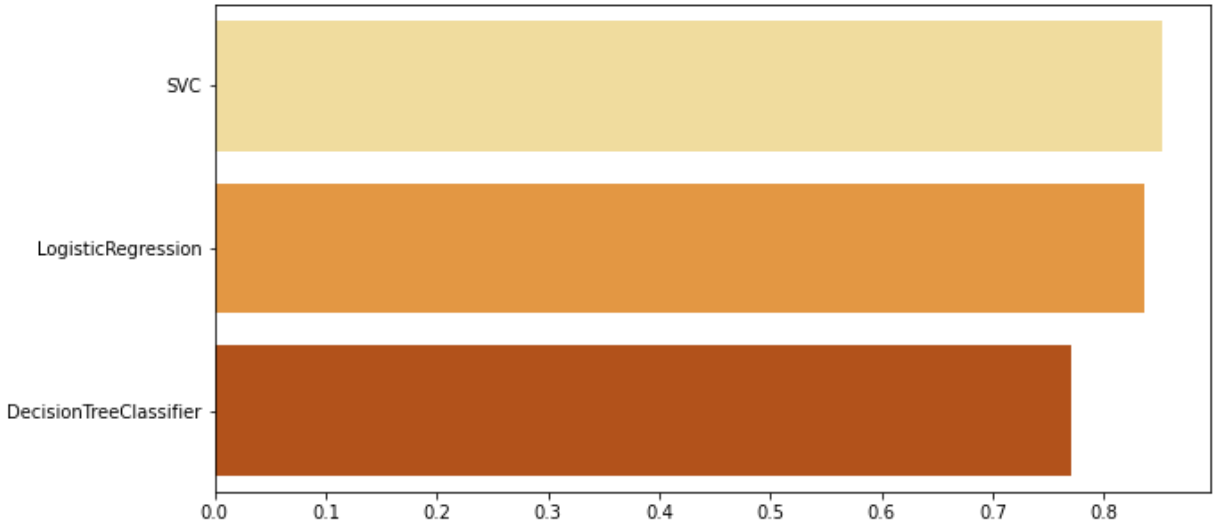
```
              precision    recall  f1-score   support

           0       0.71      0.86      0.78        29
           1       0.85      0.69      0.76        32

    accuracy                           0.77        61
   macro avg       0.78      0.77      0.77        61
weighted avg       0.78      0.77      0.77        61
```

In [37]:
```
plt.figure(figsize = (10,5))
sns.barplot(x = predicted, y = model, palette='YlOrBr')
```

Out[37]: <AxesSubplot:>



According to the graph above, it is very clear that the accuracy score is highest for SVM. So we can say that SVM is the best algorithm for this dataset. That is, the SVM model gives the most accurate prediction, when compared to the other two algorithms.

Hyper-parameter Tuning

We use a function technique called GridSearchCV for hyperparameter tuning. GridSearchCV is a technique to search through the best parameter values from the given set of the grid of parameters. It is basically a cross-validation method. the model and the parameters are required to be fed in. Best parameter values are extracted and then the predictions are made.

In [39]:
```
from sklearn.model_selection import GridSearchCV
```

In [63]:
```
parameters = [{'C': [0.1,1, 10, 100,1000],'kernel': ['linear', 'poly', 'rbf'
               'shrinking': [True,False]}]

grid_search = GridSearchCV(estimator = clf,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 13,
                           n_jobs = -1)

grid_search.fit(X_train, y_train)
best_accuracy_log = grid_search.best_score_
best_parameters = grid_search.best_params_
print(best_accuracy_log)
print(best_parameters)
```

```
0.8630229419703105
{'C': 10, 'degree': 1, 'kernel': 'poly', 'shrinking': True}

C:\Users\SRIDHAR\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: Da
taConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().
```

After Adjusting the hyper-parameter, the accuracy of the model increased by 1%.

Giving User-Input Provision for the model

In [59]:

```
age = int(input('Enter Your age : '))
print('IF you are Male Press 1 or if  Female Press 0')
sex = int(input('Female or Male : ' ))
print(' Type 0 for Typical Angina, 1 for Atypical Angina, 2 for Non-anginal
cp = int(input('Type of chest pain : '))
trtbps = int(input('Resting blood pressure (in mm Hg) : '))
chol = int(input('Cholestoral in mg/dl fetched via BMI sensor : '))
print('Fasting blood sugar > 120 mg/dl) Type 1 if True, 0 if False ')
fbs = int(input('Fasting blood sugar : '))
print('Rest ECG - 0 = Normal, 1 = ST-T wave normality, 2 = Left ventricular
restecg = int(input('Resting electrocardiographic results : '))
thalachh = int(input('Maximum heart rate achieved : '))
print('1 = Yes, 0 = No')
exng = int(input('Exercise induced angina : '))
oldpeak = float(input('Previous peak : '))
print('Type between 1 to 3')
slp = int(input('Slope : '))
caa = int(input('Number of major vessels : '))
print('Type between 0 to 3')
thall = int(input('Thalium Stress Test result : '))

X_pred = pd.DataFrame([[age,sex,cp,trtbps,chol,fbs,restecg,thalachh,exng,old
X_pred[["age","trtbps","chol","thalachh","oldpeak"]] = scaler.transform(X_pr

```

```
Enter Your age : 23
IF you are Male Press 1 or if  Female Press 0
Female or Male : 1
 Type 0 for Typical Angina, 1 for Atypical Angina, 2 for Non-anginal Pain, 3 fo
r Asymptomatic
Type of chest pain : 1
Resting blood pressure (in mm Hg) : 123
Cholestoral in mg/dl fetched via BMI sensor : 123
Fasting blood sugar > 120 mg/dl) Type 1 if True, 0 if False
Fasting blood sugar : 1
Rest ECG - 0 = Normal, 1 = ST-T wave normality, 2 = Left ventricular hypertroph
y
Resting electrocardiographic results : 1
Maximum heart rate achieved : 123
1 = Yes, 0 = No
Exercise induced angina : 1
Previous peak : 2.3
Type between 1 to 3
Slope : 2
Number of major vessels : 2
Type between 0 to 3
Thalium Stress Test result : 1
```

In [64]:
```
pred = grid_search.predict(X_pred)[0]
if(pred == 0):
    print('Our model predicted that this is most likely not a Heart Attack')
else:
    print('Our model predicts that there is a chance that this is a heart At
```

```
Our model predicted that this is most likely not a Heart Attack
```

CONCLUSION:

The best model for this dataset is SVM and the parameters which gives more accuracy is {'C': 10, 'degree': 1, 'kernel': 'poly', 'shrinking': True}. We achieved an accuracy of 86.3%. And we provided an user input provision for the users to test the model.