```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('bank _M.csv')
df.head()

df.describe()

df.isnull().sum()

df.info()

df['deposit'].value_counts()

sns.countplot(data=df, x ='deposit')
plt.show()

plt.figure(figsize=(25,18) )
sns.countplot(x=df['job'], y=None, hue=df['deposit'])
plt.show()

sns.countplot(x=df['marital'], y=None,  hue=df['deposit'])
plt.show()

df['education'].value_counts()

sns.countplot(x=df['education'], y=None, hue=df['deposit'])
plt.show()

sns.countplot(x=df['housing'], y=None, hue=df['deposit'])
plt.show()

df.corr(numeric_only= True)

sns.heatmap(df.corr(numeric_only= True), annot=True)
plt.show()

le = LabelEncoder()
df['job'] = le.fit_transform(df['job'])
df['marital'] = le.fit_transform(df['marital'])
df['education'] = le.fit_transform(df['education'])
df['default'] = le.fit_transform(df['default'])
df['housing'] = le.fit_transform(df['housing'])
df['loan'] = le.fit_transform(df['loan'])
df['contact'] = le.fit_transform(df['contact'])
df['month'] = le.fit_transform(df['month'])
df['day'] = le.fit_transform(df['day'])
df['deposit'] = le.fit_transform(df['deposit'])
df['poutcome'] = le.fit_transform(df['poutcome'])
```

```
x = df.drop('deposit', axis=1)
y= df['deposit']

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_state=42)

x_train.shape, x_test.shape, y_train.shape, y_test.shape

# Decision Tree
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(x_train, y_train)
DT.score(x_test, y_test)*100

y_pred = DT.predict(x_test)
y_pred

from sklearn.metrics import confusion_matrix
cmf = confusion_matrix(y_test, y_pred)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()

# Logistic Regression
from sklearn.linear_model import LogisticRegression
LG = LogisticRegression()
LG.fit(x_train, y_train)
LG.score(x_test, y_test)*100

ypred = LG.predict(x_test)
cmf = confusion_matrix(y_test, ypred)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()

# Support Vetor Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(x_train, y_train)
svm.score(x_test, y_test)*100

y_predict = svm.predict(x_test)
cmf = confusion_matrix(y_test, y_predict)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()

# KNN
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier()
KNN.fit(x_train, y_train)
KNN.score(x_test, y_test)*100

ypredict = KNN.predict(x_test)
cmf = confusion_matrix(y_test, ypredict)
```

```
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()

# Random Forest
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()
RF.fit(x_train, y_train)
RF.score(x_test, y_test)*100

yPredict = RF.predict(x_test)
cmf = confusion_matrix(y_test, yPredict)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()

from sklearn.metrics import precision_score
print("Precision Score of Decision Tree :",precision_score(y_test, y_pred))
print("Precision Score of Logistic Regression :",precision_score(y_test, ypred))
print("Precision Score of Support Vetor Machine :",precision_score(y_test,
y_predict))
print("Precision Score of KNN :",precision_score(y_test, ypredict))
print("Precision Score of Random Forest :",precision_score(y_test, yPredict))

from sklearn.metrics import recall_score
print("Recall of Decision Tree :",round(recall_score(y_test, y_pred), 2))
print("Recall of Logistic Regression :",round(recall_score(y_test, ypred), 2))
print("Recall of Support Vetor Machine :",round(recall_score(y_test, y_predict),
2))
print("Recall of KNN :",round(recall_score(y_test, ypredict), 2))
print("Recall of Random Forest :",round(recall_score(y_test, yPredict), 2))
from sklearn.metrics import accuracy_score
print("Accuracy Score of Decision Tree :",accuracy_score(y_test, y_pred))
print("Accuracy Score of Logistic Regression :",accuracy_score(y_test, ypred))
print("Accuracy Score of Support Vetor Machine :",accuracy_score(y_test,
y_predict))
print("Accuracy Score of KNN :",accuracy_score(y_test, ypredict))
print("Accuracy Score of Random Forest :",accuracy_score(y_test, yPredict))

from sklearn.metrics import classification_report
sc = classification_report(y_test, y_pred)

df.columns

def subscripation(age, job, marital, education, default, balance, housing,loan,
contact, day, month, duration, campaign, pdays, previous, poutcome):
    ax = np.array([age, job, marital, education, default, balance, housing,loan,
contact, day, month, duration, campaign, pdays, previous, poutcome])
    prediction = DT.predict(ax.reshape(1, -1))
    if prediction == 0:
        return 'Not Subscribed'
    else:
        return "Subsribed"

subscripation(59, 0, 1, 1, 0, 2343, 1, 0, 2, 4, 8, 1042, 1, -1, 0, 3)
```

subscripation(33, 1, 2, 0, 0, 1, 1, 0, 0, 19, 0, 257, 1, 1, 0, 3,)