```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
#Importing necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, accuracy_score, confusion_m

path='/content/drive/MyDrive/prodigy ds/bank-additional.csv'
bank = pd.read_csv(path,sep=';')
```

```
bank.head()
```

	age	job	marital	education	default	housing	loan
0	30	blue-collar	married	basic.9y	no	yes	no
1	39	services	single	high.school	no	no	no
2	25	services	married	high.school	no	yes	no
3	38	services	married	basic.9y	no	unknown	unknown
4	47	admin.	married	university.degree	no	yes	no

5 rows × 21 columns

```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             4119 non-null   int64
 1   job             4119 non-null   object
 2   marital         4119 non-null   object
 3   education       4119 non-null   object
 4   default         4119 non-null   object
 5   housing         4119 non-null   object
 6   loan            4119 non-null   object
 7   contact         4119 non-null   object
 8   month           4119 non-null   object
 9   day_of_week     4119 non-null   object
 10  duration        4119 non-null   int64
 11  campaign        4119 non-null   int64
 12  pdays           4119 non-null   int64
 13  previous        4119 non-null   int64
 14  poutcome        4119 non-null   object
 15  emp.var.rate    4119 non-null   float64
```

```
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   object
dtypes: float64(5), int64(5), object(11)
memory usage: 675.9+ KB
```

```
bank.describe()
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.con
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.0
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704	-40.4
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349	4.5
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.8
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.7
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.8
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.4
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000	-26.9

```
bank.isnull().sum()
```

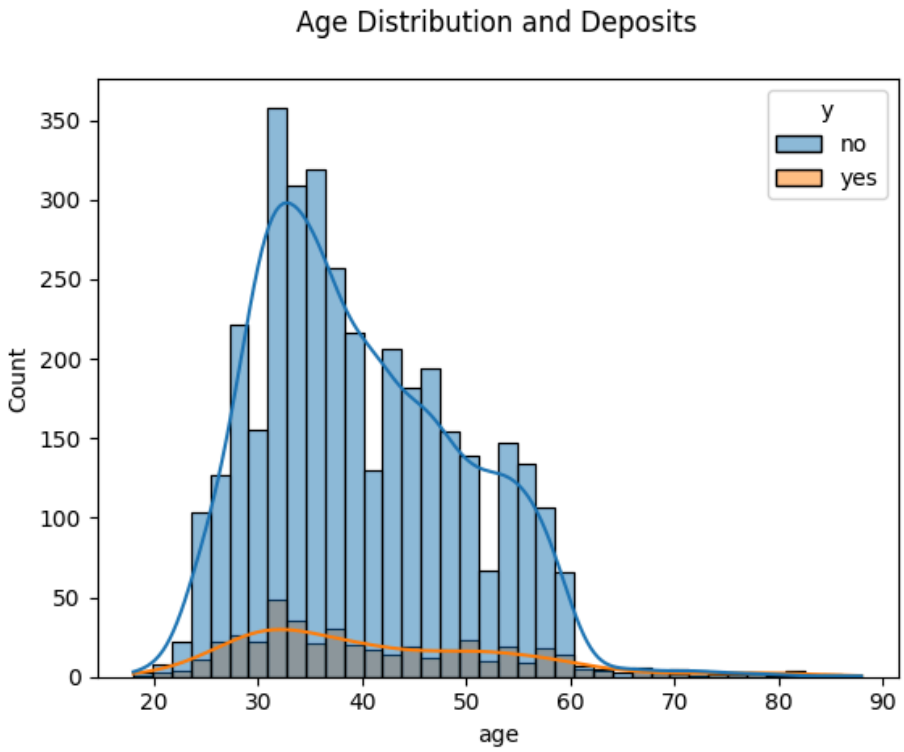
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

```
#Checking for duplicates
bank.duplicated().sum()
```

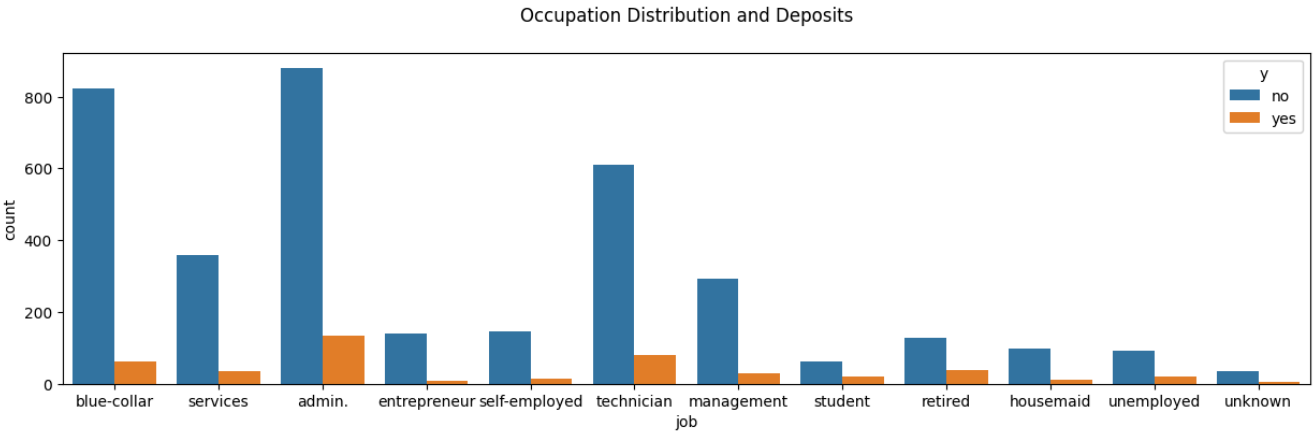
```
0
```

Exploratory Data Analysis

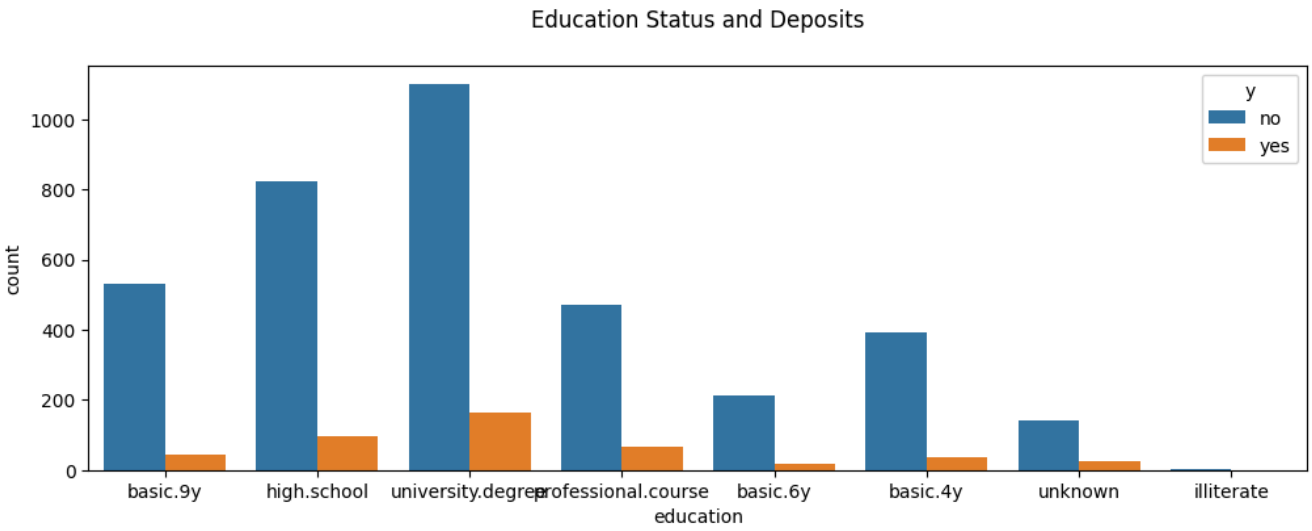
```
sns.histplot(x="age", data=bank, kde=True, hue= "y")
plt.title("Age Distribution and Deposits\n")
plt.show()
```




```
plt.figure(figsize=(15,4))
sns.countplot(x="job", data= bank, hue ="y")
plt.title("Occupation Distribution and Deposits\n")
plt.show()
```



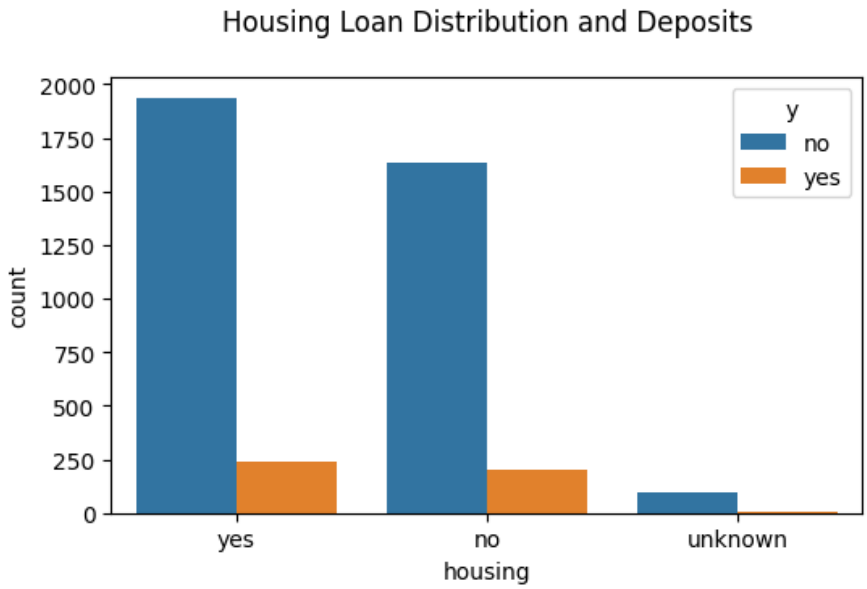

```
plt.figure(figsize=(12,4))
sns.countplot(x="education", data= bank, hue ="y")
plt.title("Education Status and Deposits\n")
plt.show()
```




```
bank.default.value_counts()
```

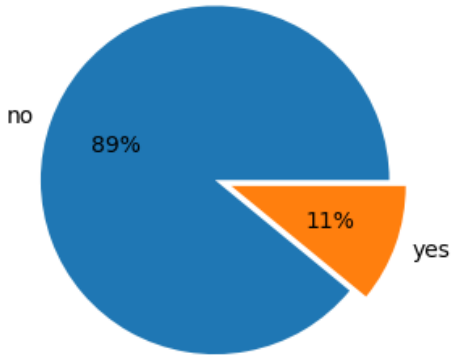
```
default
no         3315
unknown     803
yes           1
Name: count, dtype: int64
```

```
plt.figure(figsize=(6,3.5))
sns.countplot(x="housing", data= bank, hue ="y")
plt.title("Housing Loan Distribution and Deposits\n")
plt.show()
```

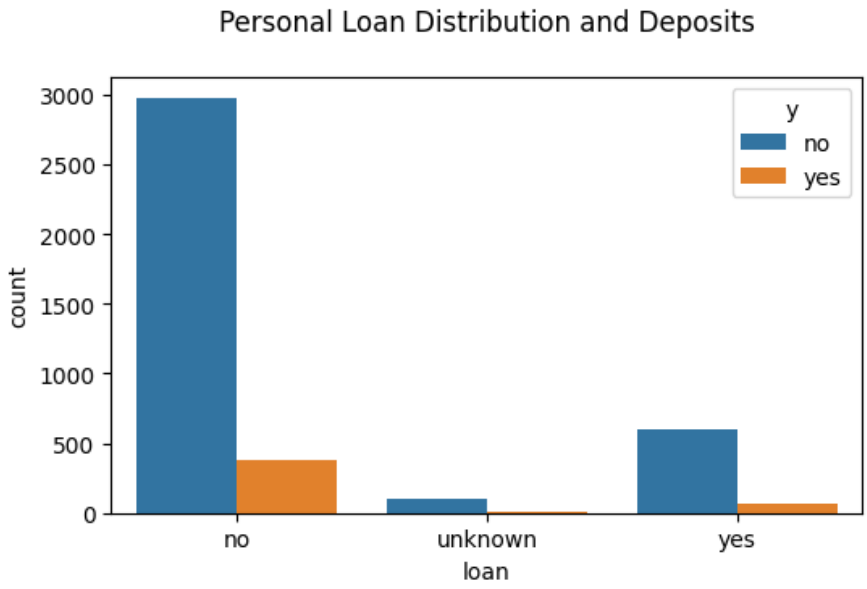



```
bank.y.value_counts()
keys = bank.y.value_counts().index
data = bank.y.value_counts().values
plt.figure(figsize=(6,3.5))
explode = [0,0.1]
plt.pie(data,labels=keys,explode=explode, autopct='%.0f%%')
```

```
plt.show()
```




```
plt.figure(figsize=(6,3.5))
sns.countplot(x="loan", data= bank, hue ="y")
plt.title("Personal Loan Distribution and Deposits\n")
plt.show()
```




```
cols = bank.select_dtypes("object").columns
cols
```

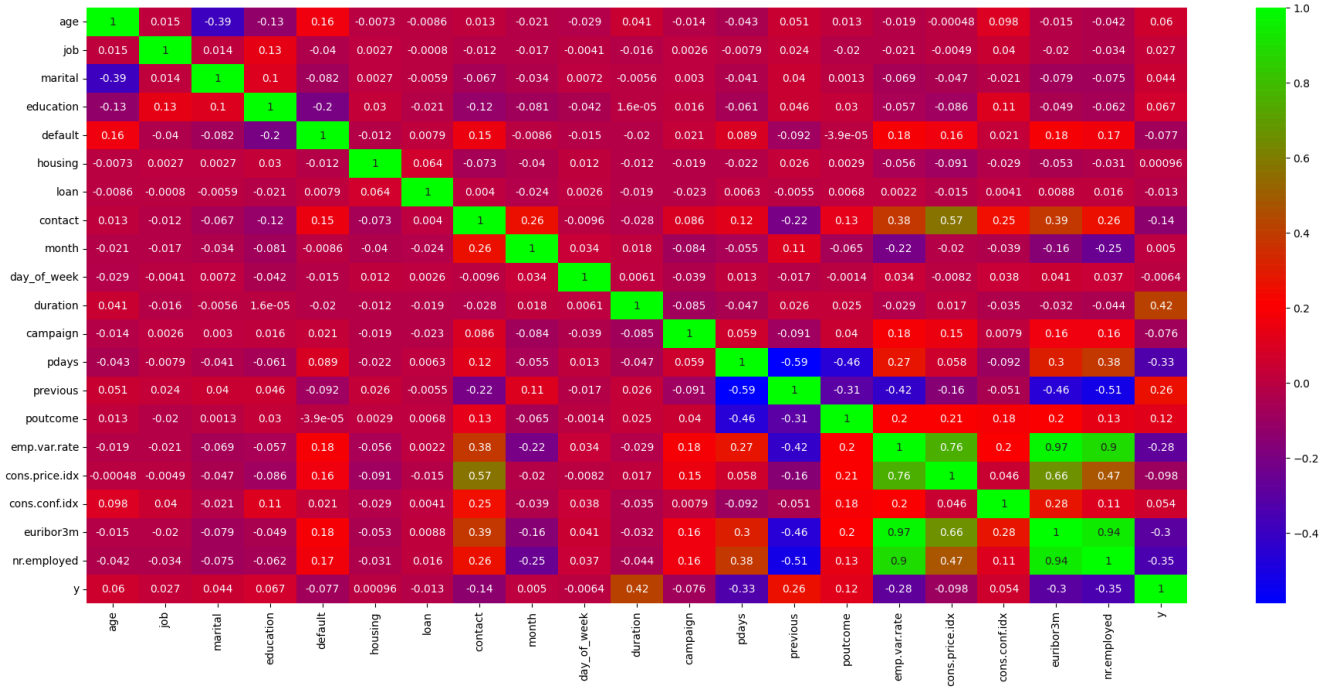
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'day_of_week', 'poutcome', 'y'],
      dtype='object')
```

```
le = LabelEncoder()

bank[cols] = bank[cols].apply(le.fit_transform)


bank.head(3)
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	p
0	30	1	1	2	0	2	0	0	6	0	...	2	999	
1	39	7	2	3	0	0	0	1	6	0	...	4	999	
2	25	7	1	3	0	2	0	1	4	4	...	1	999	

3 rows × 21 columns

```
plt.figure(figsize=(23,10))
sns.heatmap(bank.corr(), cmap='brg', annot=True)
plt.show()
```




```
X = bank.drop("y", axis=1)
y = bank.y
scaler = StandardScaler()

X_scaled = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)

#Train-test split
train_X, test_X, train_y, test_y = train_test_split(X_scaled, y, test_size=0.3
decision_tree = DecisionTreeClassifier()
decision_tree.fit(train_X, train_y)
```

```
DecisionTreeClassifier()
```

```
print('Train Score: {}'.format(decision_tree.score(train_X, train_y)))
print('Test Score: {}'.format(decision_tree.score(test_X, test_y)))
```

```
Train Score: 1.0
Test Score: 0.8818770226537217
```

```
cross_val_score(decision_tree, train_X, train_y, cv=5).mean()
```

```
0.8796432697862506
```

```
ypred = decision_tree.predict(test_X)
print(classification_report(test_y,ypred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.93      0.93      1107
           1       0.44      0.46      0.45       129

    accuracy                           0.88      1236
   macro avg       0.69      0.69      0.69      1236
weighted avg       0.88      0.88      0.88      1236
```

```
param_grid = {
    'max_depth': [3, 5, 7,10, None],
    'criterion' : ['gini', 'entropy'],
    'min_samples_leaf': [3, 5, 7, 9,10,20]
    }
gscv = GridSearchCV(decision_tree, param_grid, cv=5, verbose=1)
gscv.fit(train_X, train_y)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
```

```
GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

```
gscv.best_params_
```

```
{'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 20}
```

```
gscv.best_estimator_
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf=20)
```

```
clf = DecisionTreeClassifier(criterion= 'gini', max_depth= 5, min_samples_leaf
clf.fit(train_X, train_y)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, min_samples_leaf=3)
```

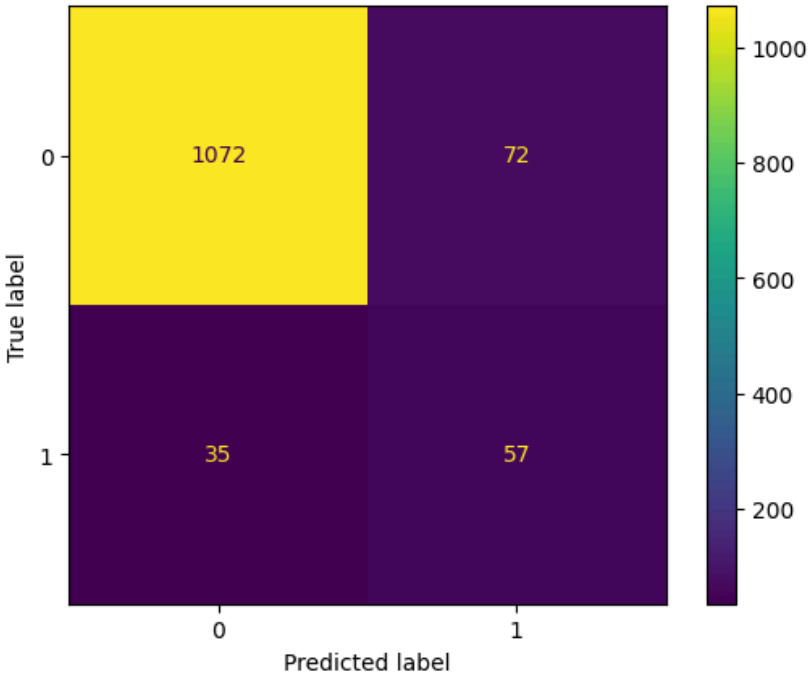
```
print('Train Score: {}'.format(clf.score(train_X, train_y)))
print('Test Score: {}'.format(clf.score(test_X, test_y)))
```

```
Train Score: 0.9299340964273326
Test Score: 0.9134304207119741
```

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

Confusion Matrix

```
cm = confusion_matrix(pred_y, test_y)
ConfusionMatrixDisplay(cm, display_labels=clf.classes_).plot()
plt.show()
```




```
print(classification_report(pred_y, test_y))
```

```
              precision    recall  f1-score   support

           0       0.97      0.94      0.95      1144
           1       0.44      0.62      0.52        92

    accuracy                           0.91      1236
   macro avg       0.71      0.78      0.73      1236
weighted avg       0.93      0.91      0.92      1236
```

```
accuracy = accuracy_score(test_y,pred_y)
print("Test Accuracy of Decision Tree Classifier : {}".format(accuracy*100))
```

```
Test Accuracy of Decision Tree Classifier : 91.34304207119742
```

```
Cross_val = cross_val_score(clf, test_X,test_y, cv=5).mean()
print("Cross-Validation Accuracy Scores Decision Tree : ",Cross_val*100)
```

The Tree Visualization

```
from sklearn import tree
fig = plt.figure(figsize=(25,20))
t= tree.plot_tree(clf,filled=True,feature_names=X.columns)
```