In [1]:
```
# Standart Imports
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn import preprocessing

# Models
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

In [6]:
```
df = pd.read_csv("/Users/liza/Desktop/IDS400 A&L/Assignments/bank.csv", sep=";")
df.head()
```

Out[6]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	c
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	

In [7]:
```
# Preprocessing
df['loan'] = df['loan'].replace(['no', 'yes'], [0, 1])
df['marital'] = df['marital'].replace(['single', 'married', 'divorced'], [0, 1,
df["default"] = df["default"].replace({"no": 0, "yes": 1})
df = pd.get_dummies(df, columns=["marital", "education", "housing"])
df['y'] = df['y'].replace(['no', 'yes'], [0, 1])
```

In [8]:
```
# Check the number of nulls in all columns that we have (print True/False if any
print(df.isnull().any())

# check for null or NaN values in all columns that we have (print True/False if
print(df.isna().any())

# Check which columns have string values
string_cols = [col for col in df.columns if df[col].dtype == 'object']
for col in string_cols:
    empty_str_count = (df[col] == '').sum()
    if empty_str_count > 0:
        print(f"Column {col} has {empty_str_count} empty strings.")
    else:
        print(f"Column {col} does not have any empty strings.")
```

```
age                   False
job                   False
default               False
```

```
balance                 False
loan                    False
contact                 False
day                     False
month                   False
duration                False
campaign                False
pdays                   False
previous                False
poutcome                False
y                       False
marital_0               False
marital_1               False
marital_2               False
education_primary       False
education_secondary     False
education_tertiary      False
education_unknown       False
housing_no              False
housing_yes             False
dtype: bool
age                     False
job                     False
default                 False
balance                 False
loan                    False
contact                 False
day                     False
month                   False
duration                False
campaign                False
pdays                   False
previous                False
poutcome                False
y                       False
marital_0               False
marital_1               False
marital_2               False
education_primary       False
education_secondary     False
education_tertiary      False
education_unknown       False
housing_no              False
housing_yes             False
dtype: bool
Column job does not have any empty strings.
Column contact does not have any empty strings.
Column month does not have any empty strings.
Column poutcome does not have any empty strings.
```

In [9]:

```
# Features columns
features = ['marital_0', 'marital_1', 'marital_2', 'education_primary', 'educati
            'education_tertiary', 'education_unknown', 'default', 'housing_no',
            'loan', 'duration', 'campaign', 'pdays']
# Target column
target = 'y'
```

In [10]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix

# Split the data into training and testing sets
train_set, test_set = train_test_split(df[features + [target]], test_size=0.2, r
X_train = train_set.drop(target, axis=1)
y_train = train_set[target]
X_test = test_set.drop(target, axis=1)
y_test = test_set[target]

# Preprocessing here is really important because we don't want to have any odd r
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Linear Model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
lin_reg_score = lin_reg.score(X_test, y_test)
print("Linear Model score:", lin_reg_score)

# Logit Model
log_reg = LogisticRegression(max_iter=1000) # because it is not coverging to the
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)
log_reg_score = log_reg.score(X_test, y_test)
print("Logistic Model score:", log_reg_score)

# Decision Tree
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, y_train)
dt_clf_score = dt_clf.score(X_test, y_test)
print("Decision Tree score:", dt_clf_score)

# KNN
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
y_pred_knn = knn_clf.predict(X_test)
knn_clf_score = knn_clf.score(X_test, y_test)
print("KNN score:", knn_clf_score)

# SVM
svm_clf = SVC()
svm_clf.fit(X_train, y_train)
svm_clf_score = svm_clf.score(X_test, y_test)
print("SVM score:", svm_clf_score)


# Print Confusion Matrix for each model
# NOTE: Confusion Matrix for Linear Model cannot be calculated because we have m
print("Confusion Matrix for Linear Model cannot be calculated because we have mi
print("Confusion Matrix for Logistic Model:")
print(confusion_matrix(y_test, log_reg.predict(X_test)))
print("Confusion Matrix for Decision Tree:")
```

```
print(confusion_matrix(y_test, dt_clf.predict(X_test)))
print("Confusion Matrix for KNN:")
print(confusion_matrix(y_test, knn_clf.predict(X_test)))
print("Confusion Matrix for SVM:")
print(confusion_matrix(y_test, svm_clf.predict(X_test)))
```

```
Linear Model score: 0.1732894668621665
Logistic Model score: 0.8939226519337017
Decision Tree score: 0.8419889502762431
KNN score: 0.8883977900552487
SVM score: 0.8961325966850828
Confusion Matrix for Linear Model cannot be calculated because we have mix of bi
nary and continuous targets.
Confusion Matrix for Logistic Model:
[[789  18]
 [ 78  20]]
Confusion Matrix for Decision Tree:
[[729  78]
 [ 65  33]]
Confusion Matrix for KNN:
[[778  29]
 [ 72  26]]
Confusion Matrix for SVM:
[[801   6]
 [ 88  10]]
```

Answer: The best model is obviously SVM because its score is the highest among all other models