CLASSWORK

In [35]:
```
import pandas as pd

# Load the dataset
cc_apps = pd.read_csv("cc_approvals.data", header=None)
from sklearn.model_selection import train_test_split

print(cc_apps.corr())

#Drop the features 11 and 13
cc_apps = cc_apps.drop([11, 13], axis=1)

# Split into train and test sets
cc_apps_train, cc_apps_test = train_test_split(cc_apps, test_size=0.33, rand
# Import numpy
import numpy as np

# Replace the '?'s with NaN in the train and test sets
cc_apps_train = cc_apps_train.replace('?', np.NaN)
cc_apps_test = cc_apps_test.replace('?', np.NaN)
cc_apps_train.fillna(cc_apps_train.mean(), inplace=True)
cc_apps_test.fillna(cc_apps_train.mean(), inplace=True)
for col in cc_apps_train.columns: # Iterate over each column of cc_apps_trai

    if cc_apps_train[col].dtypes == 'object': # Check if the column is of ob
        # Impute with the most frequent value
        # The value_counts() function returns a Series that contain counts c
        # descending order so that its first element will be the most freque
        cc_apps_train = cc_apps_train.fillna(cc_apps_train[col].value_counts
        cc_apps_test = cc_apps_test.fillna(cc_apps_train[col].value_counts()

cc_apps_train = pd.get_dummies(cc_apps_train)
cc_apps_test = pd.get_dummies(cc_apps_test)
cc_apps_test = cc_apps_test.reindex(columns=cc_apps_train.columns, fill_valu
from sklearn.preprocessing import MinMaxScaler

# Segregate features and labels into separate variables
X_train, y_train = cc_apps_train.iloc[:, :-1].values, cc_apps_train.iloc[:,
X_test, y_test = cc_apps_test.iloc[:, :-1].values, cc_apps_test.iloc[:, [-1]

# Instantiate MinMaxScaler and use it to rescale X_train and X_test
scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX_train = scaler.fit_transform(X_train)
rescaledX_test = scaler.transform(X_test)
# Import LogisticRegression
from sklearn.linear_model import LogisticRegression

# Instantiate a LogisticRegression classifier with default parameter values
logreg = LogisticRegression()

# Fit logreg to the train set
logreg.fit(rescaledX_train,y_train)
# Import confusion_matrix
from sklearn.metrics import confusion_matrix

# Use logreg to predict instances from the test set and store it
y_pred = logreg.predict(rescaledX_test)

# Get the accuracy score of logreg model and print it
```

```
print("Accuracy of logistic regression classifier: ", logreg.score(rescaledX

# Print the confusion matrix of the logreg model
confusion_matrix(y_test,y_pred)
```

```
          2         7         10        14
2   1.000000  0.298902  0.271207  0.123121
7   0.298902  1.000000  0.322330  0.051345
10  0.271207  0.322330  1.000000  0.063692
14  0.123121  0.051345  0.063692  1.000000
Accuracy of logistic regression classifier:  1.0
```

```
/var/folders/4j/bnvctt7152z615l6szd4m7wh0000gn/T/ipykernel_97515/1641907923.
py:20: FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this will rais
e TypeError.  Select only valid columns before calling the reduction.
  cc_apps_train.fillna(cc_apps_train.mean(), inplace=True)
/var/folders/4j/bnvctt7152z615l6szd4m7wh0000gn/T/ipykernel_97515/1641907923.
py:21: FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this will rais
e TypeError.  Select only valid columns before calling the reduction.
  cc_apps_test.fillna(cc_apps_train.mean(), inplace=True)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/valid
ation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
  y = column_or_1d(y, warn=True)
```

```
Out[35]: array([[103,   0],
                [  0, 125]])
```

```
In [47]: # # Import GridSearchCV
# from sklearn.model_selection import GridSearchCV

# # Define the grid of values for tol and max_iter
# tol = [0.01, 0.001 ,0.0001]
# max_iter = [100, 150, 200]

# # Create a dictionary where tol and max_iter are keys and the lists of the
# param_grid = dict(tol=tol, max_iter=max_iter)
# # Instantiate GridSearchCV with the required parameters
# grid_model = GridSearchCV(estimator=logreg, param_grid=param_grid, cv=5)

# # Fit data to grid_model
# grid_model_result = grid_model.fit(rescaledX, y)

# # Summarize results
# best_score, best_params = grid_model_result.best_score_, grid_model_result
# print("Best: %f using %s" % (best_score, best_params))
```

TASK 1

```
In [50]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
cc_apps = pd.read_csv("cc_approvals.data", header=None)

# Drop the features 11 and 13
```

```python
cc_apps = cc_apps.drop([11, 13], axis=1)

# Split into train and test sets
cc_apps_train, cc_apps_test = train_test_split(cc_apps, test_size=0.33, rand

# Replace the '?'s with NaN in the train and test sets
cc_apps_train = cc_apps_train.replace('?', np.NaN)
cc_apps_test = cc_apps_test.replace('?', np.NaN)

# Fill NaN values with the mean of the training set
cc_apps_train.fillna(cc_apps_train.mean(), inplace=True)
cc_apps_test.fillna(cc_apps_train.mean(), inplace=True)

# Fill missing categorical values with the most frequent value
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
cc_apps = pd.read_csv("cc_approvals.data", header=None)

# Drop the features 11 and 13
cc_apps = cc_apps.drop([11, 13], axis=1)

# Split into train and test sets
cc_apps_train, cc_apps_test = train_test_split(cc_apps, test_size=0.33, rand

# Replace the '?'s with NaN in the train and test sets
cc_apps_train = cc_apps_train.replace('?', np.NaN)
cc_apps_test = cc_apps_test.replace('?', np.NaN)

# Fill NaN values with the mean of the training set
cc_apps_train.fillna(cc_apps_train.mean(), inplace=True)
cc_apps_test.fillna(cc_apps_train.mean(), inplace=True)

# Fill missing categorical values with the most frequent value
for col in cc_apps_train.columns:
    if cc_apps_train[col].dtypes == 'object':
        cc_apps_train = cc_apps_train.fillna(cc_apps_train[col].value_counts
        cc_apps_test = cc_apps_test.fillna(cc_apps_train[col].value_counts()


# Perform one-hot encoding on categorical features
cc_apps_train = pd.get_dummies(cc_apps_train)
cc_apps_test = pd.get_dummies(cc_apps_test)
# Make sure the test set has the same columns as the train set
cc_apps_test = cc_apps_test.reindex(columns=cc_apps_train.columns, fill_valu

# Segregate features and labels into separate variables
X_train, y_train = cc_apps_train.iloc[:, :-1].values, cc_apps_train.iloc[:,
X_test, y_test = cc_apps_test.iloc[:, :-1].values, cc_apps_test.iloc[:, [-1]

# Instantiate MinMaxScaler and use it to rescale X_train and X_test
scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX_train = scaler.fit_transform(X_train)
rescaledX_test = scaler.transform(X_test)

# Find the best value of 'k' using a loop
best_k = None
best_accuracy = 0
```

```
for k in range(1, 21):  # Testing k values from 1 to 20
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(rescaledX_train, y_train)
    y_pred = knn.predict(rescaledX_test)
    accuracy = accuracy_score(y_test, y_pred)

    print(f"Accuracy for k={k}: {accuracy}")

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k

print(f"Best k: {best_k}")
print(f"Best accuracy: {best_accuracy}")
```

```
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97515/3391759154.
py:22: FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this will rais
e TypeError.  Select only valid columns before calling the reduction.
  cc_apps_train.fillna(cc_apps_train.mean(), inplace=True)
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97515/3391759154.
py:23: FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this will rais
e TypeError.  Select only valid columns before calling the reduction.
  cc_apps_test.fillna(cc_apps_train.mean(), inplace=True)
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97515/3391759154.
py:47: FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this will rais
e TypeError.  Select only valid columns before calling the reduction.
  cc_apps_train.fillna(cc_apps_train.mean(), inplace=True)
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97515/3391759154.
py:48: FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this will rais
e TypeError.  Select only valid columns before calling the reduction.
  cc_apps_test.fillna(cc_apps_train.mean(), inplace=True)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
```

```
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
```

```
Accuracy for k=1: 0.9210526315789473
Accuracy for k=2: 0.9078947368421053
Accuracy for k=3: 0.9298245614035088
Accuracy for k=4: 0.9254385964912281
Accuracy for k=5: 0.9254385964912281
Accuracy for k=6: 0.9210526315789473
Accuracy for k=7: 0.9166666666666666
Accuracy for k=8: 0.9122807017543859
Accuracy for k=9: 0.9122807017543859
Accuracy for k=10: 0.9166666666666666
Accuracy for k=11: 0.9166666666666666
Accuracy for k=12: 0.9166666666666666
Accuracy for k=13: 0.9210526315789473
Accuracy for k=14: 0.9254385964912281
Accuracy for k=15: 0.9254385964912281
Accuracy for k=16: 0.9298245614035088
Accuracy for k=17: 0.9342105263157895
Accuracy for k=18: 0.9342105263157895
Accuracy for k=19: 0.9385964912280702
Accuracy for k=20: 0.9298245614035088
Best k: 19
Best accuracy: 0.9385964912280702
```

```
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_
classification.py:215: DataConversionWarning: A column-vector y was passed w
hen a 1d array was expected. Please change the shape of y to (n_samples,), f
or example using ravel().
  return self._fit(X, y)
```

TASK 2

In [37]:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Load the dataset
cc_apps = pd.read_csv("cc_approvals.data", header=None)
cc_apps = cc_apps.drop([11, 13], axis=1)
# Replace the '?'s with NaN
cc_apps = cc_apps.replace('?', np.NaN)
cc_apps.fillna(cc_apps.mean(), inplace=True)

# Impute missing values with the most frequent value for each column
for col in cc_apps.columns:
    if cc_apps[col].dtypes == 'object':
        cc_apps = cc_apps.fillna(cc_apps[col].value_counts().index[0])

# Perform one-hot encoding on categorical features
cc_apps = pd.get_dummies(cc_apps)

# Split into features (X) and target labels (y)
X = cc_apps.iloc[:, :-1].values  # Use all columns except the last one as fe
y = cc_apps.iloc[:, -1].values   # Use the last column as the target label

# Scale the features to a range between 0 and 1
scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(X)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(rescaledX, y, test_size=

# Instantiate and train the Logistic Regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Use the trained model to make predictions on the test set
y_pred = logreg.predict(X_test)
```

```
# Calculate and print the accuracy of the Logistic Regression model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of logistic regression classifier: ", accuracy)

# Print the confusion matrix of the Logistic Regression model
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy of logistic regression classifier:  1.0
Confusion Matrix:
 [[103   0]
 [  0 125]]
```

```
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97515/2160929429.
py:13: FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this will rais
e TypeError.  Select only valid columns before calling the reduction.
  cc_apps.fillna(cc_apps.mean(), inplace=True)
```