DSC 550 - Week 9 - Exercise

In [2]:
```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.dummy import DummyRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

In [3]:
```
# 1.- Import data set
loan_df = pd.read_csv("Loan_Train.csv")
```

In [4]:
```
# 2.- Prepare data set
# Drop Loan_ID column
loan_df.drop('Loan_ID', axis=1, inplace=True)

# Drop any rows with missing data
loan_df.dropna(inplace=True)

# Convert categorical features into dummy variables
loan_df = pd.get_dummies(loan_df, columns=['Gender', 'Married',
                                           'Education','Self_Employed',
                                           'Property_Area', 'Dependents'])
```

In [5]:
```
# 3.- Split Data into training and test sets
# Set Loan_Status as target
y = loan_df['Loan_Status']

# Set remaining data set as training set
X = loan_df.drop('Loan_Status', axis=1)

# Divide training and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [6]:
```
# 4.- Create a pipeline with a min-max scaler and a KNN classifier
# Create Standardizer
standardizer = StandardScaler()

# Create a KNN classifier
knn = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)

# Create a pipeline
pipe = Pipeline([('standardizer',standardizer), ('knn',knn)])
```

In [7]:
```
# 5.- Fit a default KNN classifier to the data with the pipeline.
# Fit the pipeline
pipe.fit(X_train, y_train)
```

```
# Make predictions
y_pred = pipe.predict(X_test)

# 3. Report the model accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy on the test knn set: {accuracy:.4f}")
```

```
Model accuracy on the test knn set: 0.7708
```

In [8]:
```
# 6.- Create a search space for your KNN classifier where n_neighbors" ranges from 1 to 10.
search_space = [{"knn__n_neighbors":[1,2,3,4,5,6,7,8,9,10]}]
```

In [9]:
```
# 7.- Fit a grid search with your pipeline, search space, and 5-fold cross-validation.
classifier = GridSearchCV(pipe, search_space, cv=5, verbose=0)
```

In [10]:
```
# 8.- Find the accuracy of the grid search best model on the test set.
# Fit Classifier
classifier.fit(X_train, y_train)

# Get the best parameter and best score
best_n_neighbors = classifier.best_params_['knn__n_neighbors']
print(f"Best number of neighbors: {best_n_neighbors}")

# Evaluate the model on the test set
test_accuracy = classifier.score(X_test, y_test)
print(f"KNN classifier set accuracy: {test_accuracy:.4f}")
```

```
Best number of neighbors: 10
KNN classifier set accuracy: 0.7708
```

In [11]:
```
# 9.- repeat step 6 and 7
# Update pipeline
pipe = Pipeline([('standardizer', StandardScaler()),
                 ('classifier', KNeighborsClassifier())])

# Create dictionary with candidate learning
search_space = [
    {"classifier": [LogisticRegression(max_iter=500, solver='liblinear')],
     "classifier__penalty": ['l1','l2'],
     "classifier__C": np.logspace(0,4,10)},

    {"classifier": [RandomForestClassifier()],
     "classifier__n_estimators": [10,100,1000],
     "classifier__max_features": [1,2,3]},

    {"classifier": [KNeighborsClassifier()],
     "classifier__n_neighbors":[1,2,3,4,5,6,7,8,9,10]}
]

# Create grid search
gridsearch = GridSearchCV(pipe, search_space, cv=5, verbose=0)

# Fit grid search
best_model = gridsearch.fit(X_train, y_train)

# Print best model
print(best_model.best_estimator_)
```

```
Pipeline(steps=[('standardizer', StandardScaler()),
                ('classifier',
                 LogisticRegression(C=2.7825594022071245, max_iter=500,
                                    penalty='l1', solver='liblinear'))])
```

In [12]:
```
# 10.- Find accuracy of best model
y_pred = best_model.predict(X_test)

# Calculate and print accuracy score
accuracy = accuracy_score(y_test, y_pred)

print(f"The accuracy of the best model (Logistic Regression) is: {accuracy:.4f}")
```

```
The accuracy of the best model (Logistic Regression) is: 0.8229
```

11.- Summarize your results. This exercise illustrates the process that can be followed to test several models and parameters. This process helps to identify which model and set of parameters, out of several tested at the same time, might have the highest accuracy score. The use of GridsearchCV allows to streamline the search of multiple classifiers, models, and parameters. This exercise concludes that a Logisitc Regression with a c value of 2.78, max_iter of 500, penalty of l1, and solver equal to liblinear is the most accurate model/set of parameters with a score of 82.3%.