In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

1. Import the dataset and ensure that it loaded properly.

In [2]:
```
loan_df = pd.read_csv("Loan_Train.csv")
print(loan_df.shape)
loan_df.head(5)
```

```
(614, 13)
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	

2. Prepare the data for modeling by performing the following steps:

* Drop the column “Load_ID.”
* Drop any rows with missing data.
* Convert the categorical features into dummy variables.

In [3]:
```
# Create a new column that converts the class to a coded class of 1 and 0 (Y=1 and N=0)
loan_df['Loan_Status_Coded'] = loan_df['Loan_Status'].replace(to_replace=["Y","N"], valu
```

In [4]:
```
#2.1 Drop the column “Load_ID.”
loan_df = loan_df.drop("Loan_ID",axis=1)
loan_df.head(5)
```

Out[4]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	No	0	Graduate	No	5849	0.0	NaN
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0
4	Male	No	0	Graduate	No	6000	0.0	141.0

In [5]:
```
#2.2 Drop any rows with missing data.
loan_df = loan_df.dropna()
```

```
loan_df.head(5)
```

Out[5]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0
4	Male	No	0	Graduate	No	6000	0.0	141.0
5	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0

In [9]:
```
#2.3 Convert the categorical features into dummy variables.
loan_df = pd.get_dummies(loan_df, columns = loan_df.select_dtypes(include='object').keys
loan_df.head(5)
```

Out[9]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status_Coded	Ge
1	4583	1508.0	128.0	360.0	1.0	0	
2	3000	0.0	66.0	360.0	1.0	1	
3	2583	2358.0	120.0	360.0	1.0	1	
4	6000	0.0	141.0	360.0	1.0	1	
5	5417	4196.0	267.0	360.0	1.0	1	

5 rows × 23 columns

3. Split the data into a training and test set, where the “Loan_Status” column is the target.

In [10]:
```
x_data = loan_df.loc[:, ~loan_df.columns.isin(['Loan_Status_Y', 'Loan_Status_N'])].value
y_data = loan_df[['Loan_Status_Y']].values
```

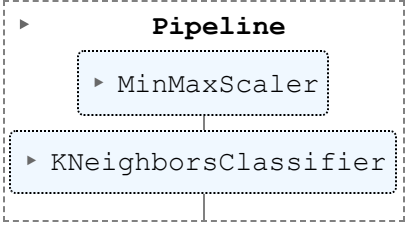
In [11]:
```
x_data = loan_df.drop(['Loan_Status_Coded', 'Loan_Status_N', 'Loan_Status_Y'], axis = 1)
# get the target
y_data = loan_df['Loan_Status_Coded'] # (Y=1 and N=0)
```

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data ,test_size = 0.2)
```

4. Create a pipeline with a min-max scaler and a KNN classifier (see section 15.3 in the Machine Learning with Python Cookbook).

In [13]:
```
#Create scaler
minmax_scale = MinMaxScaler(feature_range=(0,1))
# Scale feature
#scaled_feature = minmax_scale.fit_transform(x_train)
#scaled_feature

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

#Create a pipeline with min-max scaler
pipe= Pipeline([("scaler", minmax_scale), ("knn", knn)])
pipe
```

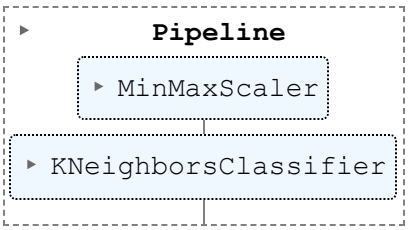
Out[13]:



5. Fit a default KNN classifier to the data with this pipeline. Report the model accuracy on the test set. Note: Fitting a pipeline model works just like fitting a regular model.

In [14]:
```
#Fitting Pipeline Model
pipe.fit(x_train,y_train.ravel())
```

Out[14]:



In [15]:
```
## Predict Output
y_pred = pipe.predict(x_test)
y_pred
```

Out[15]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0], dtype=int64)
```

In [16]:
```
#score the knn model on the testing data
pipe.score(x_test,y_test)
```

Out[16]:
```
0.7291666666666666
```

In [17]:
```
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

Out[17]:
```
0.7291666666666666
```

6. Create a search space for your KNN classifier where your "n_neighbors" parameter varies from 1 to 10. (see section 15.3 in the Machine Learning with Python Cookbook).

In [18]:
```
# Create space of candidate values
search_space = [{"knn__n_neighbors": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}]
```

7. Fit a grid search with your pipeline, search space, and 5-fold cross-validation to find the best value for the "n_neighbors" parameter.

In [19]:
```
# Create grid search
classifier = GridSearchCV(pipe, search_space, cv=5, verbose=0).fit(x_train, y_train)
```

Fitting 5 folds for each of 9 candidates, totalling 45 fits

8. Find the accuracy of the grid search best model on the test set. Note: It is possible that this will not be an improvement over the default model, but likely it will be.

In [20]:
```
knn_predict = classifier.predict(x_test)
knn_predict
```

Out[20]:
```
array([1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0], dtype=int64)
```

In [21]:
```
# Conduct nested cross-validation and outut the average score
cross_val = cross_val_score(classifier, x_data, y_data)
cross_val
```

Out[21]:
```
array([0.72916667, 0.67708333, 0.69791667, 0.66666667, 0.72916667])
```

In [22]:
```
cross_val.mean()
```

Out[22]:
```
0.7
```

In [23]:
```
#accuracy of the training dataset with tuning
# ROC-AUC score for the best model
accuracy = classifier.best_score_
accuracy
```

Out[23]:
```
0.724025974025974
```

In [24]:
```
knn_accuracy = accuracy_score(y_test,knn_predict)
knn_accuracy
```

Out[24]:
```
0.7604166666666666
```

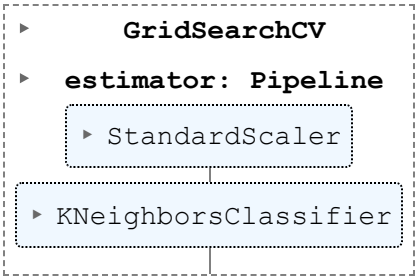
9. Now, repeat steps 6 and 7 with the same pipeline, but expand your search space to include logistic regression and random forest models with the hyperparameter values in section 12.3 of the Machine Learning with Python Cookbook.

In [25]:
```
# Create logistic regression object
logistic_regression = LogisticRegression(solver='liblinear')

# Create a pipeline (with KNeighborsClassifier without min-max scalar)
pipe= Pipeline([('standardizer', StandardScaler()), ("classifier", KNeighborsClassifier(

search_space=[
    {'classifier' : [KNeighborsClassifier()],
     'classifier__n_neighbors' : [1,2,3,4,5,6,7,8,9,10]},
    {'classifier' : [LogisticRegression()],
     'classifier__penalty' : ['l1', 'l2'],
    'classifier__C' : np.logspace(0, 4, 10),
    'classifier__solver' : ['liblinear']},
    {'classifier' : [RandomForestClassifier()],
    'classifier__n_estimators' : [10,100,1000],
    'classifier__max_features' : [1, 2, 3]}
]
```

In [26]:
```
#Create grid search and Fit grid search
classifier_best_model = GridSearchCV(pipe, search_space, cv = 5, verbose=0).fit(x_train,
```

```
classifier_best_model
```

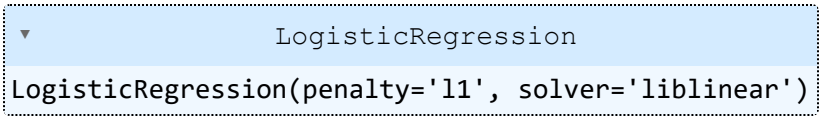
Out[26]:



10. What are the best model and hyperparameters found in the grid search? Find the accuracy of this model on the test set.

In [27]:
```
# View best model
classifier_best_model.best_estimator_.get_params()['classifier']
```

Out[27]:



In [28]:
```
classifier_best_model.best_params_
```

Out[28]:
```
{'classifier': LogisticRegression(penalty='l1', solver='liblinear'),
 'classifier__C': 1.0,
 'classifier__penalty': 'l1',
 'classifier__solver': 'liblinear'}
```

In [29]:
```
# ROC-AUC score for the best model
classifier_best_model.best_score_
```

Out[29]:
```
0.8151401230348598
```

In [30]:
```
best_model_predict = classifier_best_model.predict(x_test)
```

In [31]:
```
best_model_accuracy = accuracy_score(y_test,best_model_predict)
best_model_accuracy
```

Out[31]:
```
0.8020833333333334
```

In [32]:
```
cross_val_score(classifier, x_data, y_data)
```

Out[32]:
```
array([0.72916667, 0.67708333, 0.69791667, 0.66666667, 0.72916667])
```

In [34]:
```
cross_val_score(classifier_best_model, x_data, y_data.ravel(), cv=5)
```

Out[34]:
```
array([0.80208333, 0.78125   , 0.77083333, 0.86458333, 0.78125   ])
```

11. Summarize your results.

We analyzed Grid search with two scenarios:

1. Using KNN Classifier
2. Uisng LogisticRegression and RandomForestClassifier

- The accuracy score for the first case is 76% and second case is 80.2%.
- Best score for the first case is 72.4% and second case is 81.5%

The Logistic Regression with the hyperparameters is the best model to use for this dataset.