Classification

Agenda

In this lesson, we will cover the following concepts of classification with the help of a business use case:

- Linear vs. nonlinear classifiers
- Naive Baye's theorem
- Support vector machines
- Evaluating the model using accuracy and confusion matrix

Linear vs. Nonlinear Classifiers

Linear Classifiers

1. Linear classifiers use a linear combination of input features to categorize data into labels.
2. They use the data to establish a linear boundary.
3. They use a line, plane, or hyperplane (a plane that exists in more than two dimensions) to distinguish data.
4. They can be used to classify data that is linearly separable.
5. They can be modified to classify data that is not linearly separable.
6. Some examples of linear classifier are:

- Perceptron
- Linear regression
- Multiple linear regression
- Logistic regression
- Support vector machine (SVM) (with linear kernel)
- Linear discriminant classifier (LDC)
- Naive Baye's Theorem
- Gradient descent

Nonlinear Classifiers

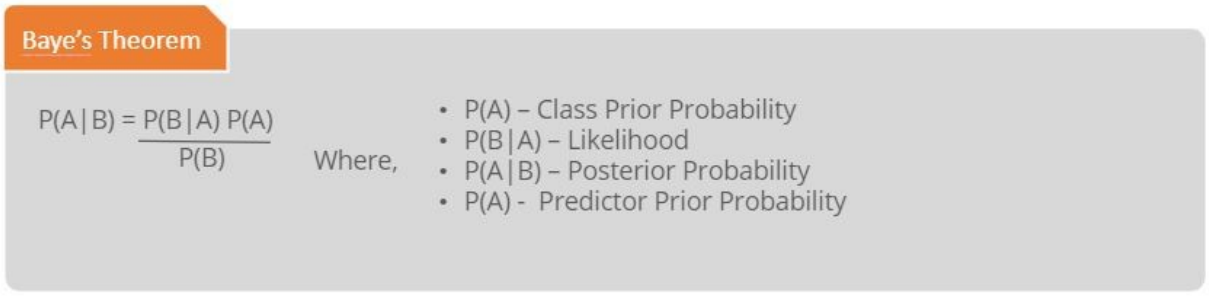
1. Nonlinear classifiers use a nonlinear combination of input features to categorize data into labels.
2. They use the data to establish a nonlinear boundary.
3. They classify data by mapping it into a high-dimensional space.
4. They are used to classify complex data which cannot be distinguished by a simple threshold or linear boundary.
5. They cannot be modified to classify data that is linearly separable.
6. Some examples of nonlinear classifiers are:

- Multilayer perceptron
- K-nearest neighbors (KNN)
- Decision trees
- Random forest
- Discriminant classifier
- Quadratic discriminant classifier (QDC)

Naive Baye's Theorem

Naive Baye's theorem is used in classifications that assume that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Baye's Theorem



Example

- As a first step toward prediction using Naive Baye's theorem, you will have to estimate the frequency of each and every attribute.

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Frequency Table		Play	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

Frequency Table		Play	
		Yes	No
Humidity	High	3	4
	Normal	6	1

Frequency Table		Play	
		Yes	No
Wind	True	3	3
	False	6	2

- Calculating the likelihood of each attribute:

Likelihood Table		Play		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$P(B|A) = P(Sunny|Yes) = 3/9 = 0.33$

$P(B) = P(Sunny) = 5/14 = 0.36$

$P(A) = P(Yes) = 9/14 = 0.64$

Similarly likelihood of "No" given Sunny is:

$P(A|B) = P(No|Sunny) = P(Sunny|No)* P(No) / P(Sunny) = (0.4 \times 0.36) /0.36 = 0.40$

Likelihood table for Humidity

Likelihood Table		Play		
		Yes	No	
Humidity	High	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

$P(Yes|High) = (3/9) * 0.64 / (7/14) = 0.42$

$P(No|High) = (4/5) * 0.36 / (7/14) = 0.58$

Likelihood table for Wind

Likelihood Table		Play		
		Yes	No	
Wind	False	6/9	2/5	8/14
	True	3/9	3/5	6/14
		9/14	5/14	

$P(Yes|Weak) = (6/9) * 0.64 / (8/14) = 0.75$

$P(No|Weak) = (2/5) * 0.36 / (8/14) = 0.25$

- Let us find the probability of playing golf under the following conditions:
  - Outlook = Rain
  - Humidity = High
  - Wind = Weak
  - Play = ?
- Solution:
  - Calculation:

Likelihood of "**Yes**" = P(Outlook = Rain|**Yes**)*P(Humidity= High|**Yes**)* P(Wind= False|**Yes**)*P(**Yes**) = 2/9 * 3/9 * 6/9 * 9/14 = 0.031

Likelihood of "**No**" = P(Outlook = Rain|**No**)*P(Humidity= High|**No**)* P(Wind= False|**No**)*P(**No**) = 3/5 * 4/5 * 2/5 * 5/14 = 0.068

  - Prediction:

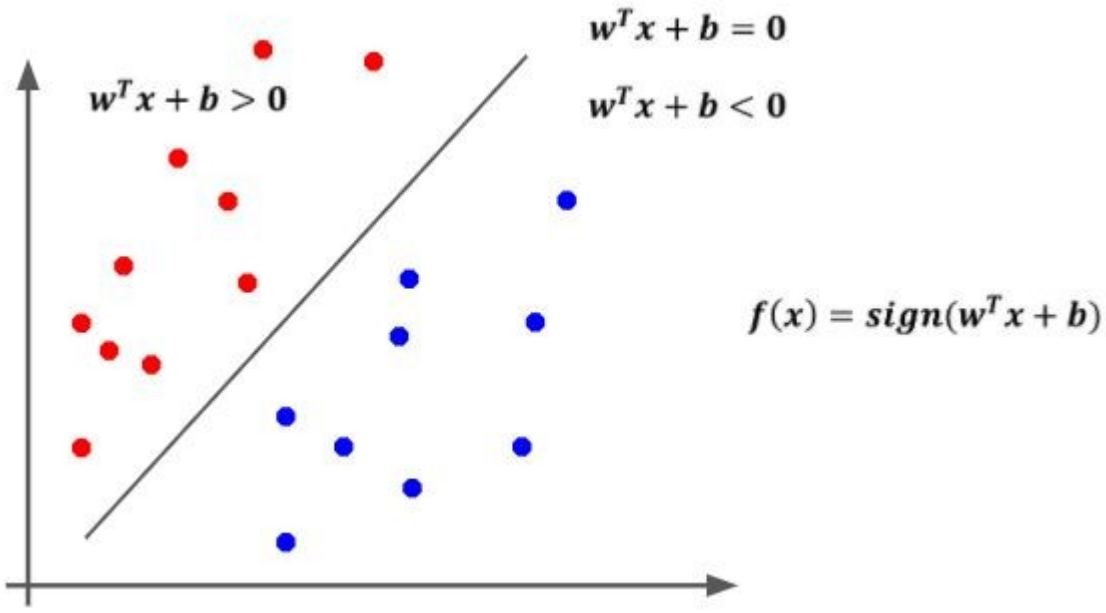
Support Vector Machine

Let us understand the following basics before moving on to the mathematics behind SVM:

- Linear Separators
- Optimal Separation
- Classification Margin

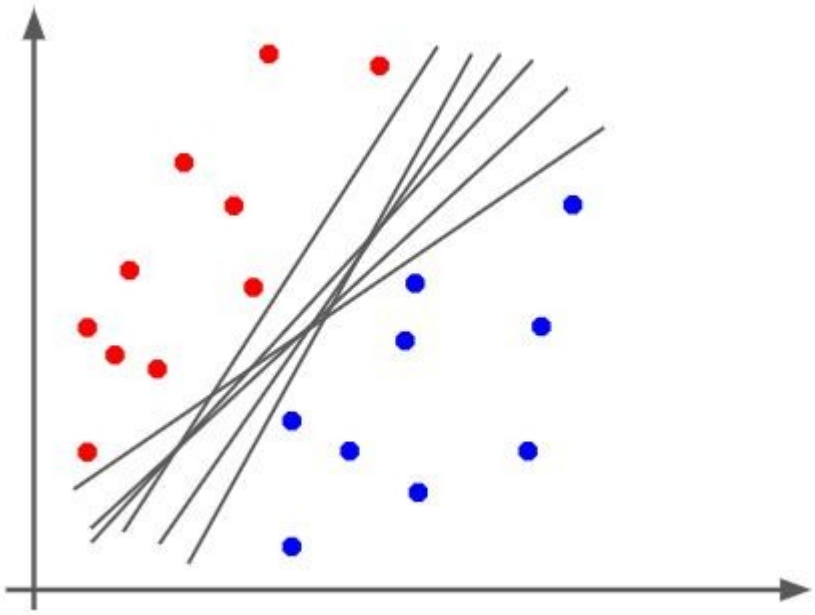
1. Linear Separators:

Consider a binary separation which can be viewed as the task of separating classes in the feature space.



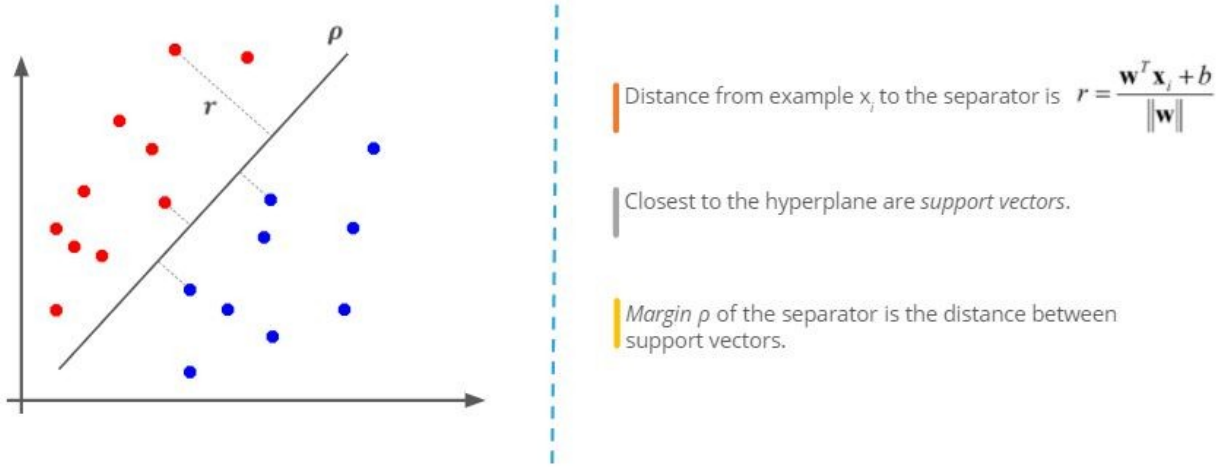
2. Optimal Separation:

Classification becomes difficult in the presence of multiple separators. Therefore, it is important to have an optimal separator.

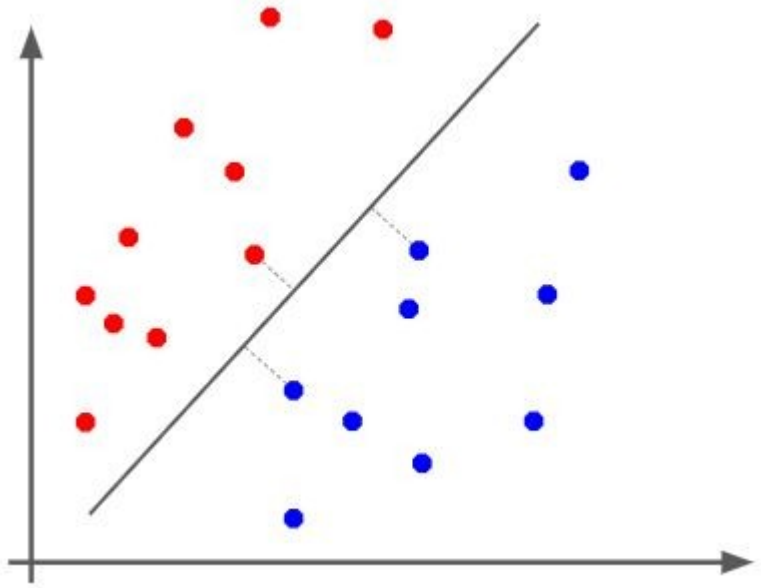


3. Classification margin:

- Concept of classification margin: Let's use the diagram below, to understand a separator and classification margin in a better manner.



- Need for maximizing the classification margin:
  - It generalizes the predictions and performs better on the test data by not overfitting the model to the training data.
  - It takes care of the support vectors, ignoring other training examples.



Linear SVM

Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\mathbf{w}^T\mathbf{x}_i + b \le -\rho/2 \quad \text{if } y_i = -1$$
$$\mathbf{w}^T\mathbf{x}_i + b \ge \rho/2 \quad \text{if } y_i = 1$$
$$\Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge \rho/2$$

For every support vector $\mathbf{x}_s$, the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, you obtain the distance between each $\mathbf{x}_s$. The hyperplane is:

$$r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

Then the margin can be expressed through (rescaled) $\mathbf{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

- Formulate the quadratic optimization problem:

Find $\mathbf{w}$ and b such that $\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized and for all $(\mathbf{x}_i, y_i)$, $i=1..n$:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1$$

- Reformulate the problem as:

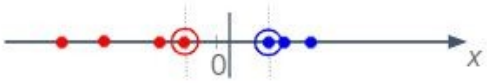
Find $\mathbf{w}$ and b such that $\Phi(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ is minimized and for all $(\mathbf{x}_i, y_i)$, $i=1..n$:

$$y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

Nonlinear SVM

Scenario 1

- Datasets that are linearly separable with some noise:

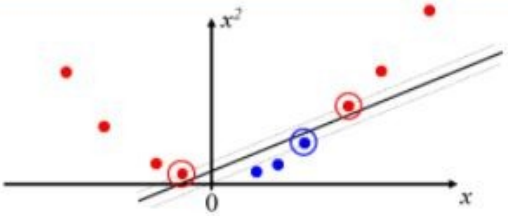


Scenario 2

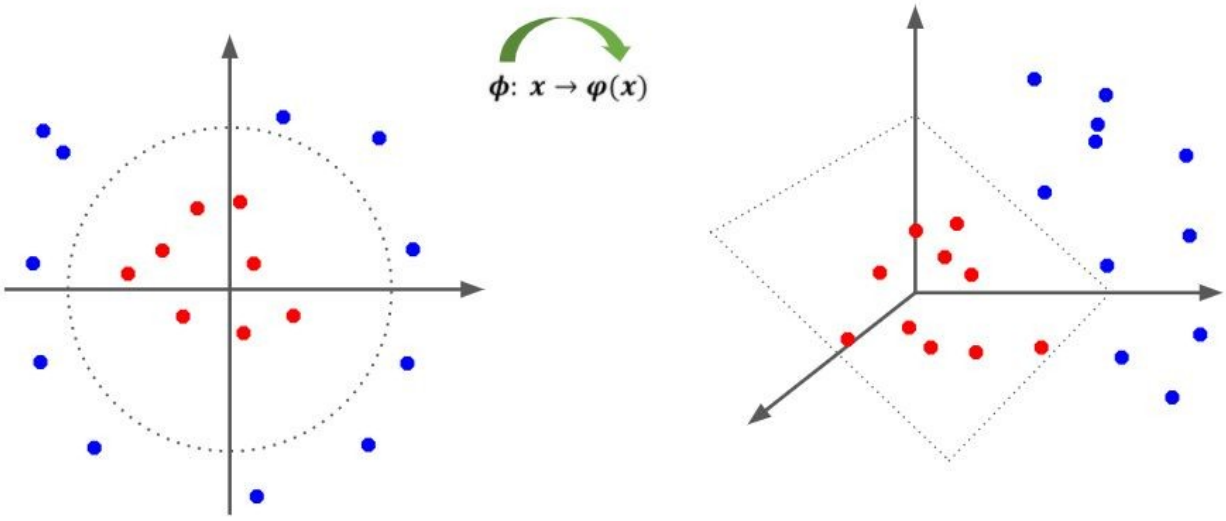
- When the dataset is hard:

Scenario 3

- Mapping data to a higher dimensional space



Feature Spaces:



The original feature space can always be mapped to some higher-dimensional feature space where the training set is separable.

Kernel Trick

- The linear classifier relies on inner product between vectors K(xi,xj)=xiTxj.
- If every datapoint is mapped into high-dimensional space via some transformation $\mathbf{\Phi: x \rightarrow \varphi(x)}$, the inner product becomes

  $\mathbf{K(xi,xj) = \varphi(xi)\, T\varphi(xj)}$
- A **kernel function** is a function that is equivalent to an inner product in a feature space.
- **Example:**

2-dimensional vectors $\mathbf{x}=[x_1\ x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2$.

Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2 = 1+x_{i1}^2x_{j1}^2+2\,x_{i1}x_{j1}x_{i2}x_{j2}+x_{i2}^2x_{j2}^2+2x_{i1}x_{j1}+2x_{i2}x_{j2}$$

$$= [1\ \ x_{i1}^2\ \sqrt{2}\,x_{i1}x_{i2}\ \ x_{i2}^2\ \sqrt{2}x_{i1}\ \sqrt{2}x_{i2}]^T[1\ \ x_{j1}^2\ \sqrt{2}\,x_{j1}x_{j2}\ \ x_{j2}^2\ \sqrt{2}x_{j1}\ \sqrt{2}x_{j2}]$$

$$= \boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_j), \quad \text{where } \boldsymbol{\phi}(\mathbf{x}) = [1\ \ x_1^2\ \sqrt{2}\,x_1x_2\ \ x_2^2\ \sqrt{2}x_1\ \sqrt{2}x_2]$$

- Thus, a kernel function implicitly maps data to a high-dimensional space (without the need to compute each φ(x) explicitly).

Use Case: Classification

Note: With the help of a use case, we will perform all the basic steps to reach the model training and prediction part.

Problem Statement

Our aim in this project is to predict if a person would buy an iPhone with respect to their gender, age, and income. We will also compare different classification algorithms..

Dataset

Before reading the data, you need to download "iphone_purchase_records.csv" dataset from the link given below and upload it to the Lab. We will use Up arrow icon which is shown in the left side under View icon. Click on the Up arrow icon and upload the file wherever it is downloaded in your system.

After this, you will see the downloaded file on the left side of your lab with all the .pynb files.

Link: https://www.dropbox.com/sh/br7x7c7gjaghibp/AAAYI4aI1u2oG4fqiVKmn-9Ua?dl=0

Solution

Note: In logistic regression, we have used PCA for feature selection, but let us take another route this time.

Import Libraries

In python, Pandas is used for data manipulation and analysis. Numpy is a package which includes a multidimensional array object as well as a number of derived objects. Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Seaborn is an open-source Python library built on top of matplotlib.

These libraries are written with the import keyword.

```
#import required libraries
import pandas as pd
from pandas import Series, DataFrame

#import required libraries for visualization
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns
```

Data Acquisition

```
# Step 1 - Load Data
import pandas as pd
data_set = pd.read_csv("./iphone_purchase_records.csv")
X = data_set.iloc[:,:-1].values
y = data_set.iloc[:, 3].values
```

```
#Preview the train data
data_set.head(1)

#Check the data type
data_set.dtypes
```

Feature Extraction

In the below code, you are using the sklearn library, which contains a lot of tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction.

1. Use LabelEncoder to convert gender to number

```
from sklearn import preprocessing
# Step 2 - Convert Gender to number
from sklearn.preprocessing import LabelEncoder
labelEncoder_gender =  LabelEncoder()
X[:,0] = labelEncoder_gender.fit_transform(X[:,0])

# Optional - if you want to convert X to float data type
import numpy as np
X = np.vstack(X[:, :]).astype(np.float)
```

Splitting Datasets

```
# Step 3 - Split data into training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=0)
```

Feature Scaling

```
# Step 4 - Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Step 5 - Logistic regression classifier
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0, solver="liblinear")
classifier.fit(X_train, y_train)

# Step 6 - Predicting logistic regression model on x_test
y_pred = classifier.predict(X_test)
```

Confusion Matrix

It can be used to find the number of correct and incorrect entries.

- If an individual has not purchased an iPhone and the expected value states that they have not purchased, it is a true negative (TN), i.e., the actual value is 0 and the predicted value is also 0.
- If an individual has not purchased an iPhone but the expected value states that they have, it is a false positive (FP), i.e., the actual value is 0 and the value expected is 1.
- If an individual has purchased an iPhone but the expected value states that they have not, it is a false negative (FN), i.e., the real value is 1 and the value expected is 0.
- If an individual has purchased an iPhone and the expected value also says that they have purchased it is True Positive (TP), i.e., the actual value is 1 and the predicted value is also 1.

Accuracy score: This is the most common metric that is used to verify model accuracy. In other words, it is the proportion of the overall number of accurate predictions to the total number of predictions.

Accuracy score = (TP+TN)/(TP+TN+FP+FN)

Recall score: It is the proportion of positive incidents that we correctly expected.

Recall score = TP/(TP+FN)

Precision score: This is the proportion of positive outcomes expected that are currently positive.

Precision score = TP/(TP+FP)

```
# Step 7 - Confusion matrix
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Accuracy score:",accuracy)
precision = metrics.precision_score(y_test, y_pred)
print("Precision score:",precision)
recall = metrics.recall_score(y_test, y_pred)
print("Recall score:",recall)
```

The model has a 91 % accuracy score, an 89 % precision score, and an 81 % recall score, indicating that it works effectively.

```
# Step 8 - Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

Model Comparison

Note: Comparing the accuracy score of all the classifier models that we used above.

```
# Step 9 - Compare classification algorithms
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

classification_models = []
classification_models.append(('Logistic Regression',
LogisticRegression(solver="liblinear")))
classification_models.append(('K Nearest Neighbor',
KNeighborsClassifier(n_neighbors=5, metric="minkowski",p=2)))
classification_models.append(('Kernel SVM', SVC(kernel =
'rbf',gamma='scale')))
classification_models.append(('Naive Bayes', GaussianNB()))
classification_models.append(('Decision Tree',
DecisionTreeClassifier(criterion = "entropy")))
classification_models.append(('Random Forest',
RandomForestClassifier(n_estimators=100, criterion="entropy")))

for name, model in classification_models:
  kfold = KFold(n_splits=10, random_state=(7), shuffle=(True))
  result = cross_val_score(model, X, y, cv=kfold, scoring='accuracy')
  print("%s: Mean Accuracy = %.2f%% - SD Accuracy = %.2f%%" % (name,
result.mean()*100, result.std()*100))
```

Conclusion

From the results, we can see that KNN and Kernel SVM have done better than the others for this particular dataset. So, we will shortlist these two for this project. This is precisely the same result that we arrived at by independently applying each of those algorithms.

Note: In this topic, we saw the use of the classification methods and in this lesson we have covered the concepts of supervised learning based on regression and classification, but in the next lesson we will be working on "Decision Trees and Random Forest".

Powered by simplilearn