Machine Learning with scikit-learn

Machine learning: computer make decisions based on data without specific programming

Unsupervised learning: uncovering hidden patterns from unlabelled data

Supervised learning: predicted values are known, predict target values of unseen data

Classification predicts the label or category of a target variable
Regression predicts continuous values

Feature = Independent variable
Target = Dependent variable

1. Data must not be missing values
2. Data in numeric format
3. Data stored in pandas DF or numpy array

Scikit-learn syntax

```
from sklearn.module import Model
model = Model()
model.fit(X, y)
predictions = model.predict(X_new)
print(predictions)
```

How to build classified of unseen data

1. build a model
2. model learns form the labelled data
3. pass unlabelled data
4. model predicts the unseen data

We can use k-Neared Neighbours (knn) for clustering

```
fr sklearn.neighbors import KNeighborsClassifier
X = churn_df[["total_day_charge", "total_eve_charge"]].values
Y = churn_df["churn"].values
print(X.shape, y.shape)

knn = KNeighborsClassifier(n_neighbors=15)
knn.fit(X, y)
```

```
X_new = np.array([[1,2], [3,4], [5,6]])
print(X_new.shape)

predictions = knn.predict(X_new)
print('Predictions:{}'.format(predictions))
```

Measuring model performance

Accuracy = correct predictions / total observations

We should split data into training set and test set, fit model on training set, and test accuracy on test set

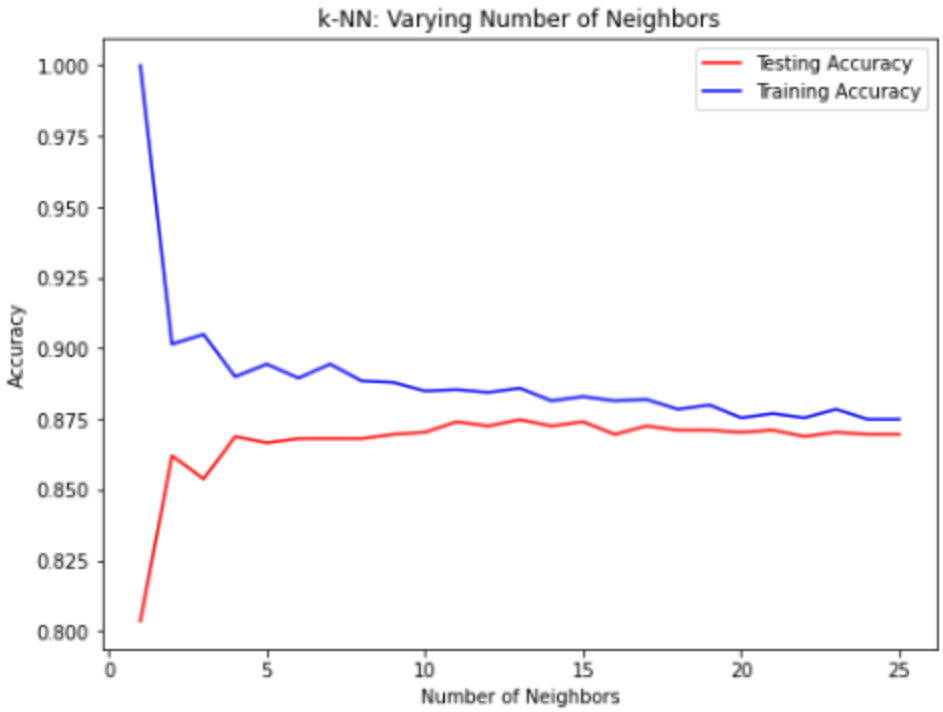
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=21, stratify=y)
knn = KNeighborsClassifier(n_neighors=6)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

As k increases, the decision boundary is less reflected on individuals

Larger k = less complex model = under fit
Smaller k = more complex model = over fit

```
train_accuracies = {}
test_accuracies = {}
neighbors = np.arange(1,26)
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors=neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

plt.figure(figsize=(8,6))
plt.title("KNN: Varying Number of Neighbors")
plt.plot(neighbors, train_accuracies.values(), label="Training Accuracy")
plt.plot(neighbors, test_accuracies.values(), label="Testing Accuracy")
plt.legend()
plt.xlabel("Number of Neighbors")
plt.ylabel("Accuracy")
plt.show()
```


k-NN: Varying Number of Neighbors
Testing Accuracy
Training Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracy
0
5
10
15
20
25
Number of Neighbors