INT 303 BIG DATA ANALYTICS

Lecture10: Models

Jia WANG

Jia.wang02@xjtlu.edu.cn

Xi'an Jiaotong-Liverpool University

西交利物浦大學

OUTLINE

- Processes Involved in Machine Learning
- Types of Machine Learning Algorithms
- Popular Algorithms with Hands On Demo
  - Linear Regression
  - Logistic Regression
  - Decision Tree and Random Forest
  - K Nearest Neighbor

PROCESSES INVOLVED IN MACHINE LEARNING

Processes involved in Machine Learning

TYPES OF MACHINE LEARNING ALGORITHMS

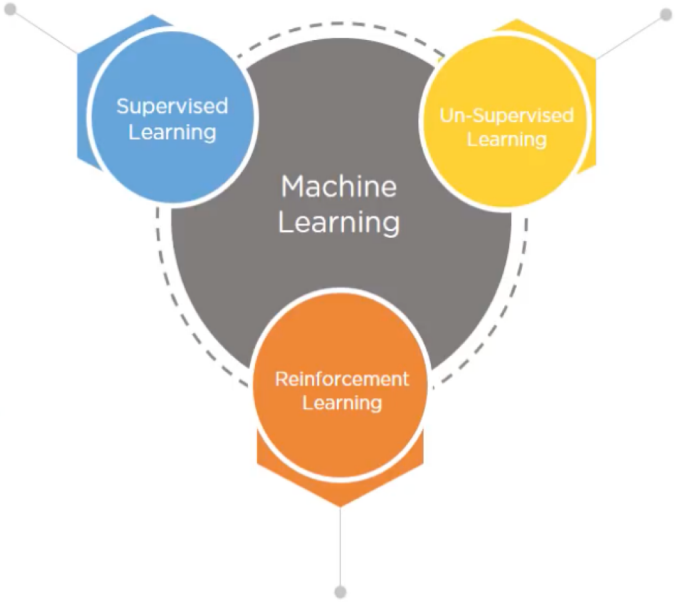
Types of Machine Learning Algorithms

Classification

- Fraud Detection
- Email Spam Detection
- Image Classification

Regression

- Weather Forecasting
- Risk Assessment
- Score Prediction



Association

- Market Basket Analysis
- Text Mining
- Face Recognition

Clustering

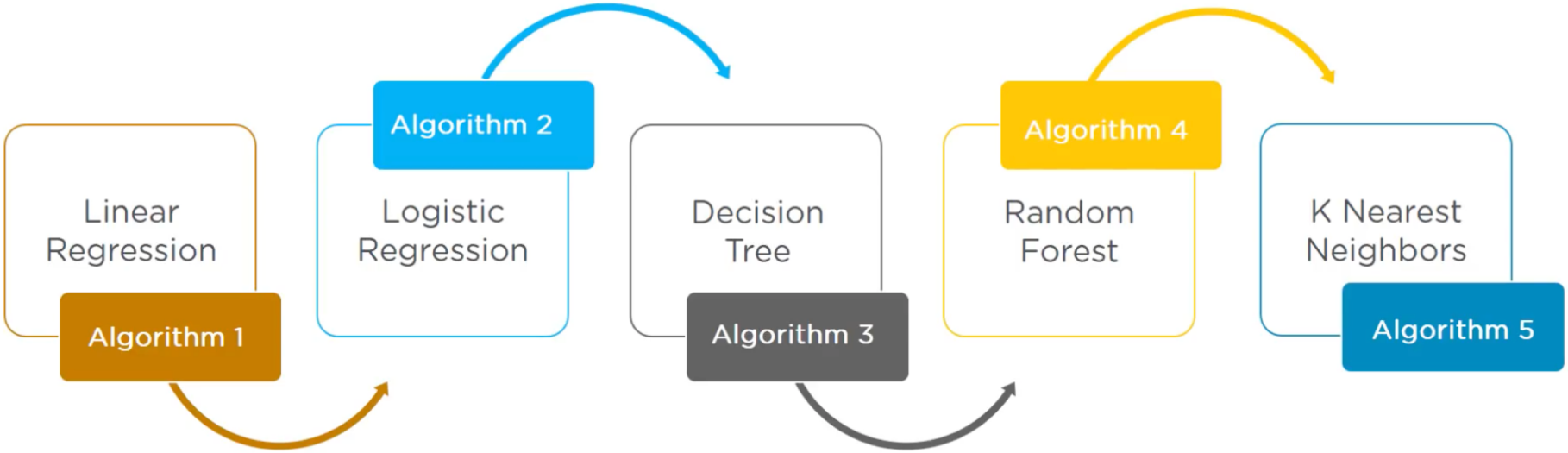
- Medical Research
- City Planning
- Targeted Marketing

Reinforcement Learning

- Gaming
- Robot Navigation
- Stock Trading
- Assembly Line Processes

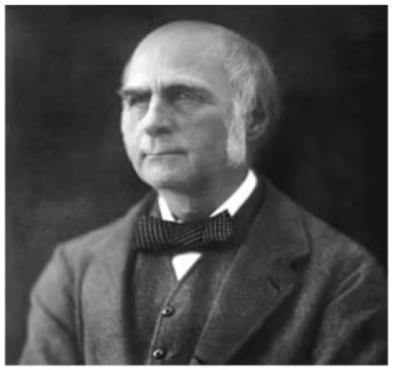
POPULAR ALGORITHMS WITH HANDS ON DEMO

Popular Algorithms in Machine Learning



History of Linear Regression

Linear Regression: Brief history on how Regression came into picture.

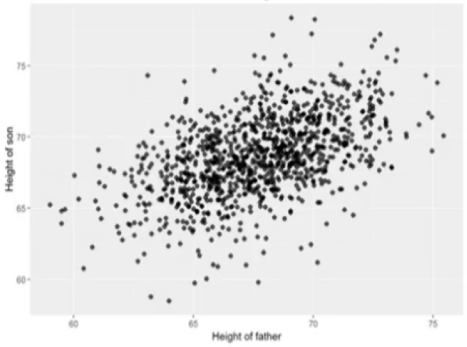


Francis Galton

Studied

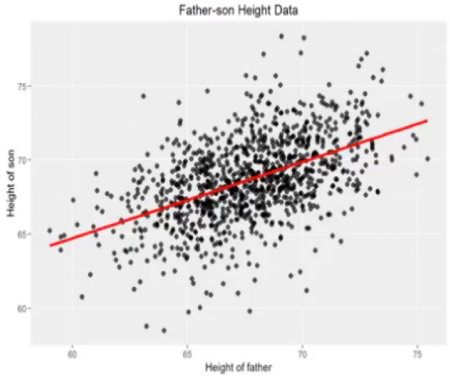
Heights of Father and Son

Analyzed



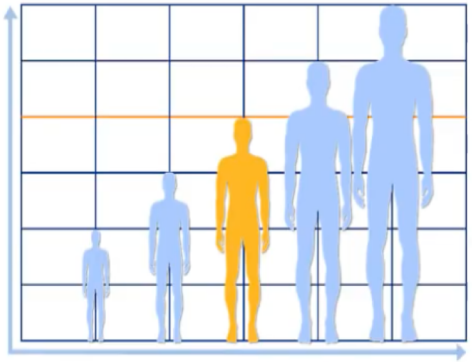
Height Data

Best fit line



Regression Line

Regressed

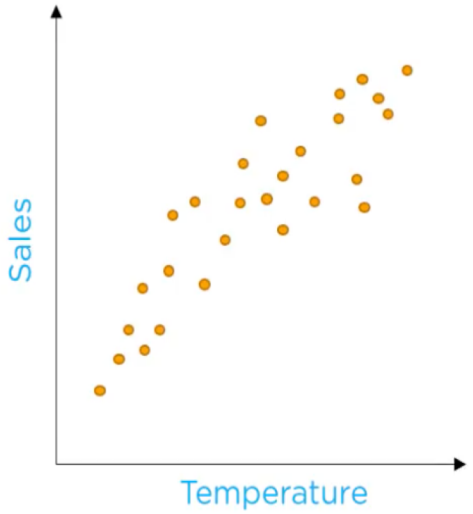


Mean Height of all People

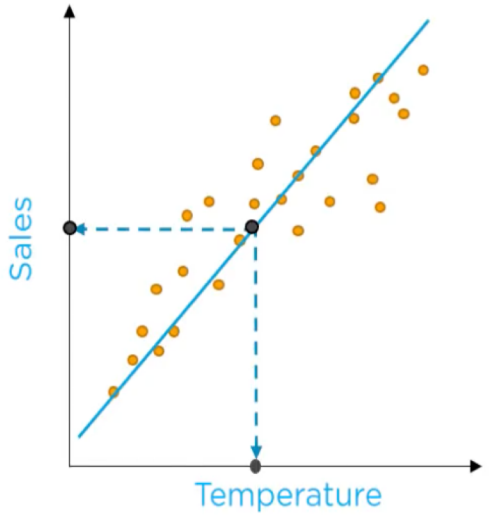
Linear Regression

Linear Regression is a linear modelling approach to find relationship between one or more independent variables (predictors) denoted as X and dependent variable (target) denoted as Y.

Predict sales of Ice Cream based on temperature:



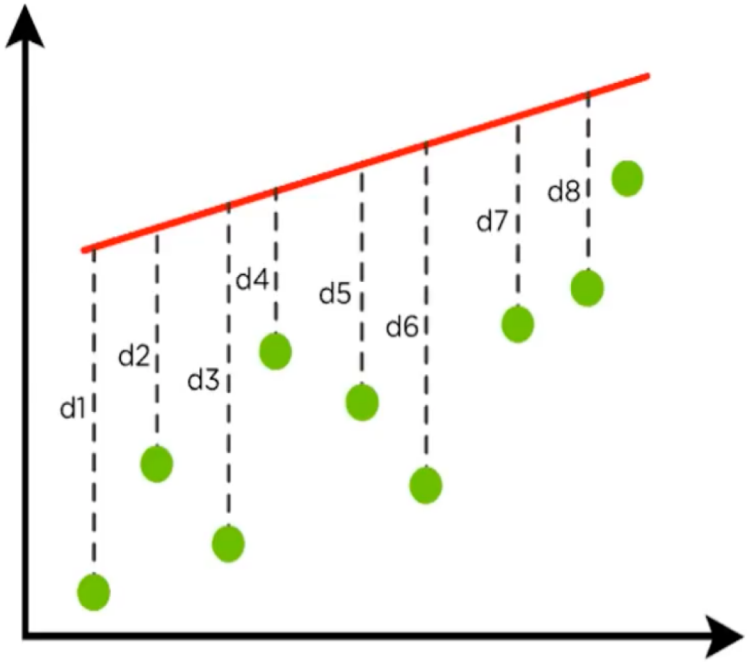
Plotting the sales of ice cream based on temperature



Regression line to predict the sales

Linear Regression

Finding the best fit line: The best fit line can be found out by minimizing the distance between all the data points and the distance to the regression line. Ways to minimize this distance are sum of squared errors, sum of absolute errors etc.



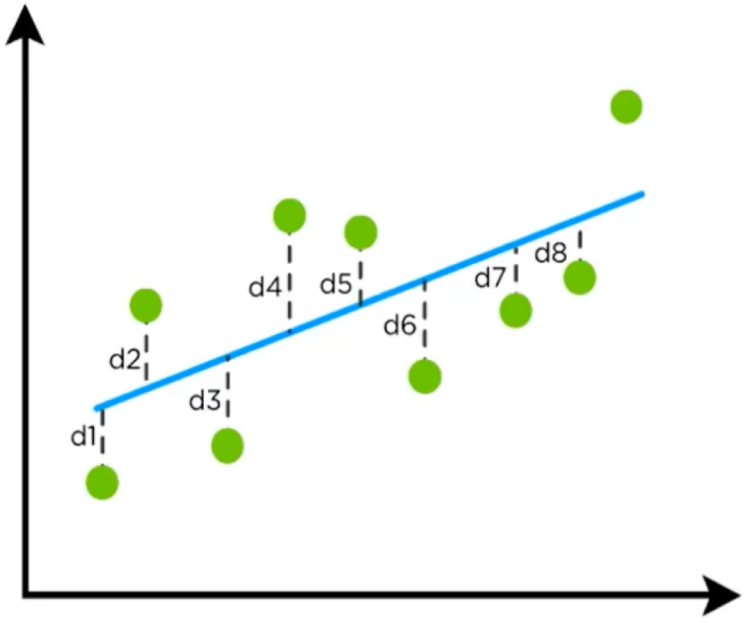
$D1 = d1^2 + d2^2 + d3^2 + d4^2 + d5^2 + d6^2 + d7^2 + d8^2 +$

Not the best fit line

Distance between the data points and the line is maximum

Linear Regression

Finding the best fit line: The best fit line can be found out by minimizing the distance between all the data points and the distance to the regression line. Ways to minimize this distance are sum of squared errors, sum of absolute errors etc.



$D = d1^2 + d2^2 + d3^2 + d4^2 + d5^2 + d6^2 + d7^2 + d8^2 +$

The regression line (blue) has the least value of D

Best fit regression line

Distance between the data points and the line is the least

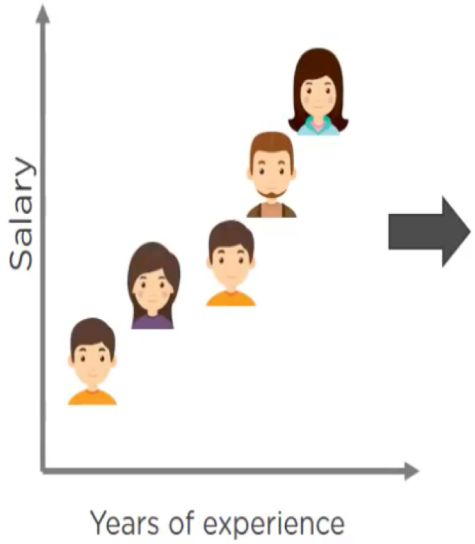
Implementation of Linear Regression

Linear Regression: Predict Employee Salary based on Years of Experience.

New Joinee

How much do we pay her?

HR team trying to figure out how much to pay to a new joinee?



Employee data of Salary based on Years of experience

Predict Salary

Implementation of Linear Regression

1. Load the libraries:

Importing the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

2. Import the Dataset:

Importing the dataset

```
salary_data = pd.read_csv('C:/Users/avijeet.biswal/Downloads/ML data sets/Salary_Data.csv')
X = salary_data.iloc[:, :-1].values
y = salary_data.iloc[:, 1].values
```

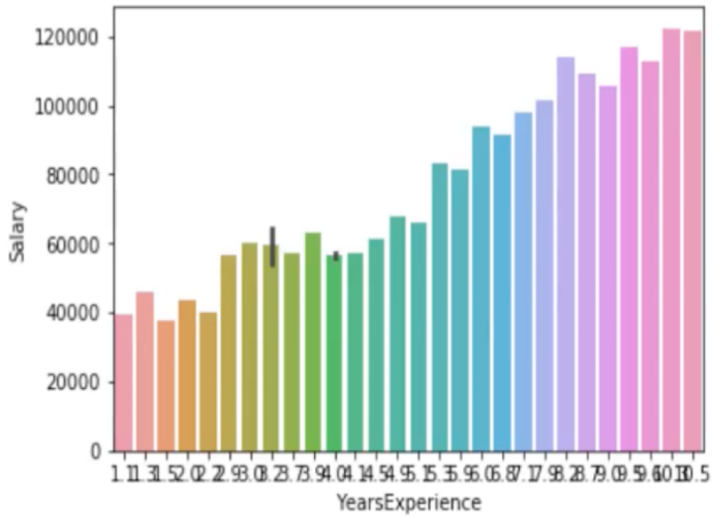
Implementation of Linear Regression

3. Visualize the data:

Visualizing the Dataset

```
sns.barplot(x='YearsExperience',y='Salary',data=salary_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xe8a9ef0>
```



Implementation of Linear Regression

4. Split the data into training and testing set:

Splitting the dataset into the Training set and 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 = 1/3, random_state = 0)
```

5. Fit Simple Linear Regression to the training dataset:

Fitting Simple Linear Regression to the Training set

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Implementation of Linear Regression

6. Predict the test set results:

Predicting the Test set results

```
y_pred = lr.predict(X_test)
y_pred
```

```
array([  40835.10590871,  123079.39940819,   65134.55626083,
         63265.36777221,  115602.64545369,  108125.8914992 ,
        116537.23969801,   64199.96201652,   76349.68719258,
        100649.1375447 ])
```

Implementation of Linear Regression

7. Visualize the train set results:

Visualising the Training set results

```
plt.scatter(X_train, y_train, color = 'blue')
plt.plot(X_train, lr.predict(X_train), color = 'red')
plt.title('Salary ~ Experience (Train set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

8. Visualize the test set results:

Visualising the Test set results

```
plt.scatter(X_test, y_test, color = 'blue')
plt.plot(X_train, lr.predict(X_train), color = 'red')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Implementation of Linear Regression

9. Calculating the residuals:

Finding the Residuals

```
# Calculating the Residuals
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test,y_pred))
print('MSE:', metrics.mean_squared_error(y_test,y_pred))
print('RMSE:', np.sqrt(metrics.mean_absolute_error(y_test,y_pred)))
```

```
MAE: 3426.42693743
MSE: 21026037.3295
RMSE: 58.5356894333
```

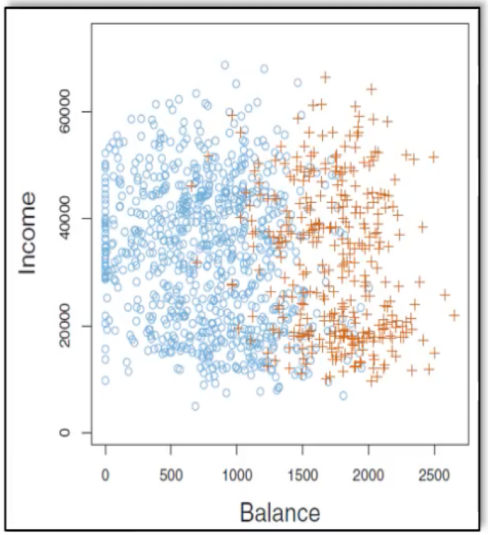
Logistic Regression

Logistic Regression is a Classification algorithm used to predict discrete/ categorical values

Who will default on their credit card payment?

Credit card users

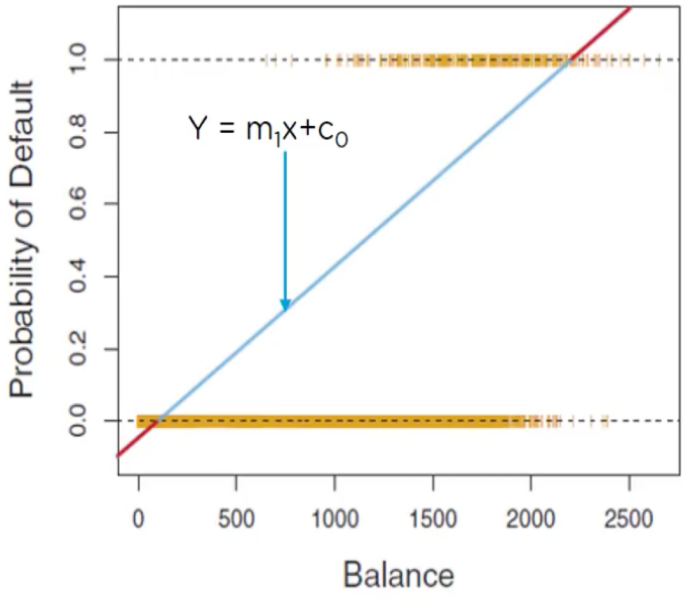
Make credit card transaction



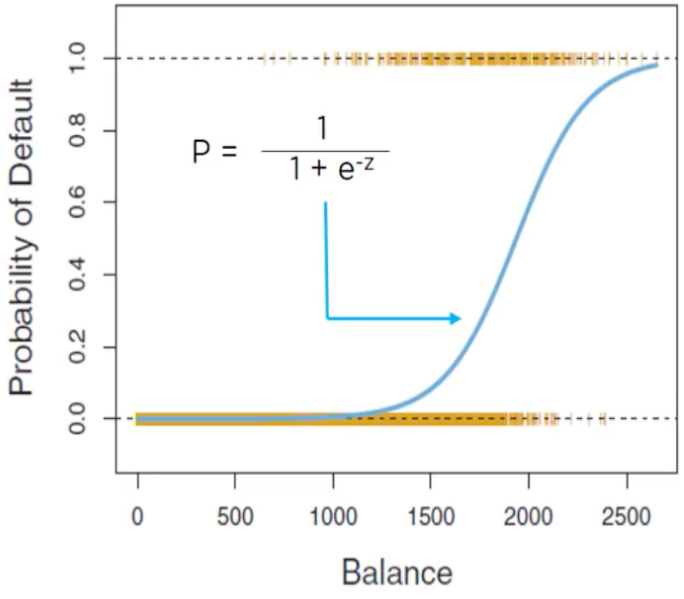
Plot of monthly credit card balance and annual income

Logistic Regression

Plotting the Logistic Regression Curve: The Logistic Regression curve is known as the Sigmoid curve (S curve)



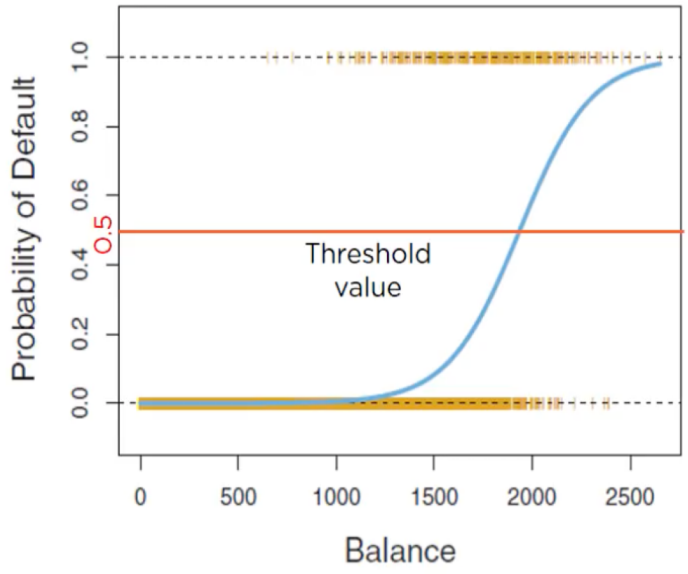
Predicted Y can exceed the range 0 and 1



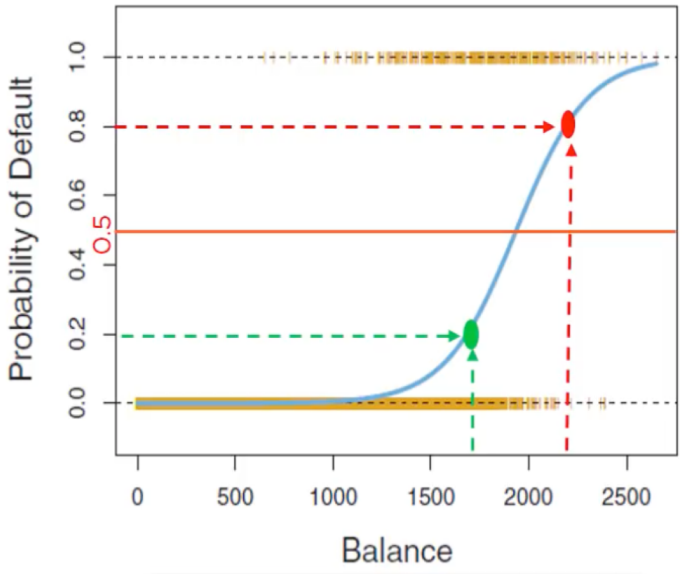
Predicted Y lies in the range 0 and 1

Logistic Regression

Plotting the Logistic Regression Curve: The Logistic Regression curve is known as the Sigmoid curve (S curve)



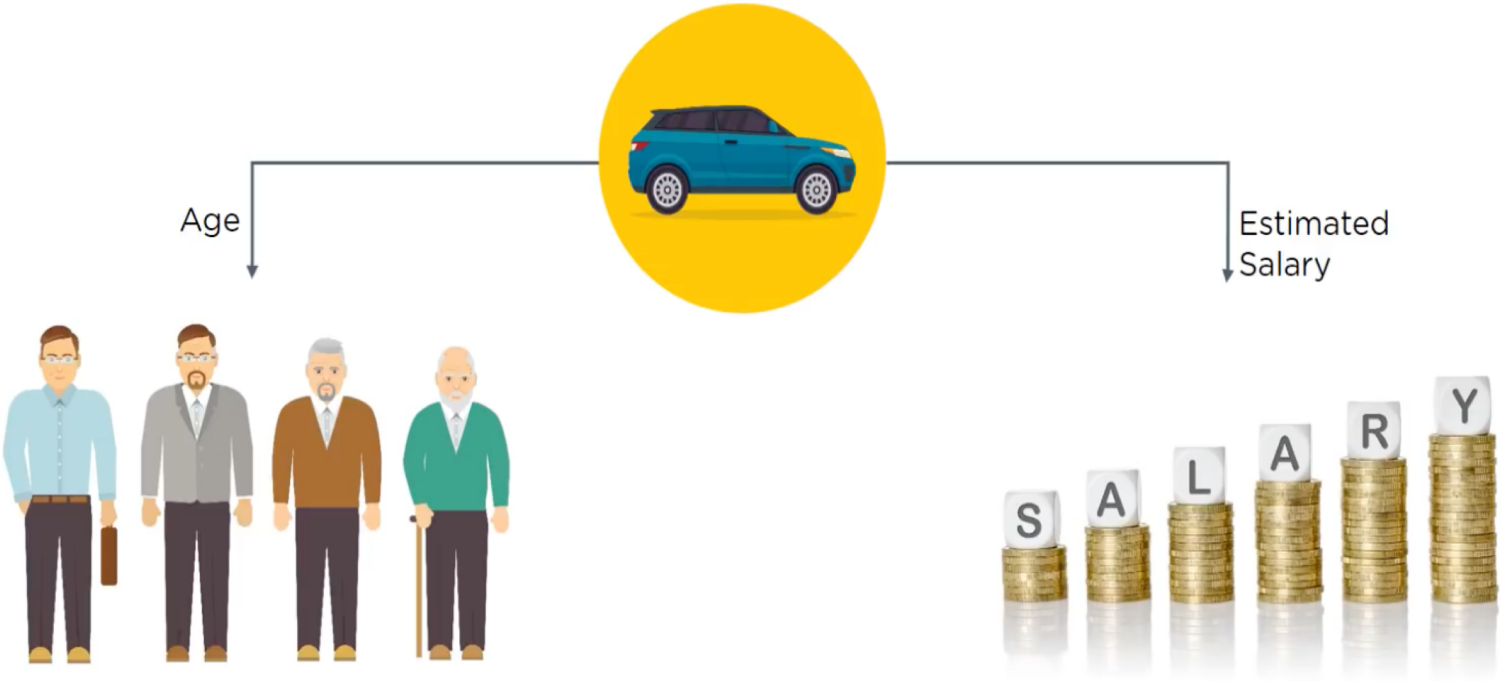
Cutoff point at 0.5, anything below it results in 0 and above is 1



Red data point will default as it is above the threshold value of 0.5 and green data point won't as it is below the threshold value

Implementation of Logistic Regression

Logistic Regression: Predict if a person will buy an SUV based on their Age and Estimated Salary



Implementation of Logistic Regression

1. Load the libraries:

Importing the Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
```

2. Import the dataset and extract the independent and dependent variables:

Importing the Dataset

```
social_network = pd.read_csv('C:/Users/avijeet.biswal/Desktop/ML data sets/SocialNetworkAds.csv')
```

Extracting the independent variables

```
X = social_network.iloc[:, [2,3]].values
```

Extracting the dependent variables

```
y = social_network.iloc[:, 4].values
```

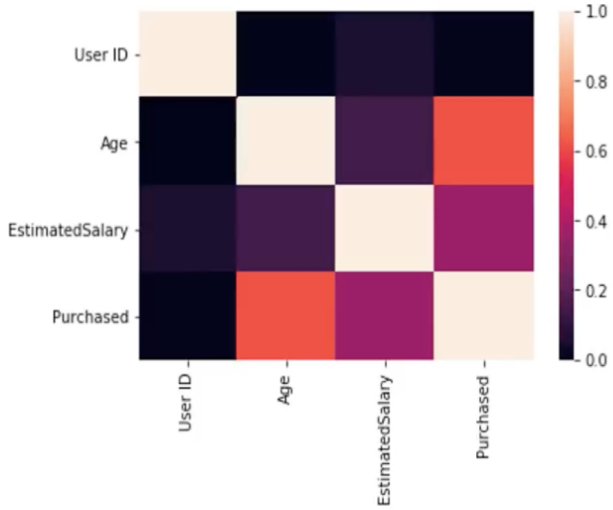
Implementation of Logistic Regression

3. Visualize the dataset:

Visualising the data by drawing a correlation map

```
sns.heatmap(social_network.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xd4152b0>
```



Implementation of Logistic Regression

4. Split the dataset into Training and Testing set:

Splitting the dataset into training and testing set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

5. Feature Scaling:

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

6. Fit Logistic Regression to Training dataset:

Fitting logistic regression to training dataset

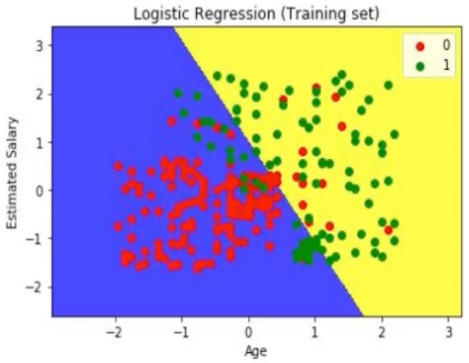
```
from sklearn.linear_model import LogisticRegression
logR = LogisticRegression(random_state = 0)
logR.fit(X_train, y_train)
```

Implementation of Logistic Regression

8. Visualize the Train set results:

Visualising the Training set results

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', 'yellow')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

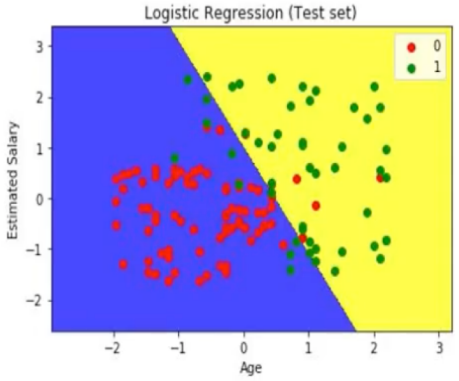


Implementation of Logistic Regression

9. Visualize the Test set results:

Visualising the Test set results

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', 'yellow')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Implementation of Logistic Regression

11. Evaluating the model:

Confusion Matrix to evaluate the model

```
# Evaluate the model using confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[79,  6],
       [11, 38]], dtype=int64)
```

Understanding the Confusion Matrix:

N = 134	Predicted: No	Predicted: Yes
Actual: No	TN=79	FP=6
Actual: Yes	FN=11	TP=38

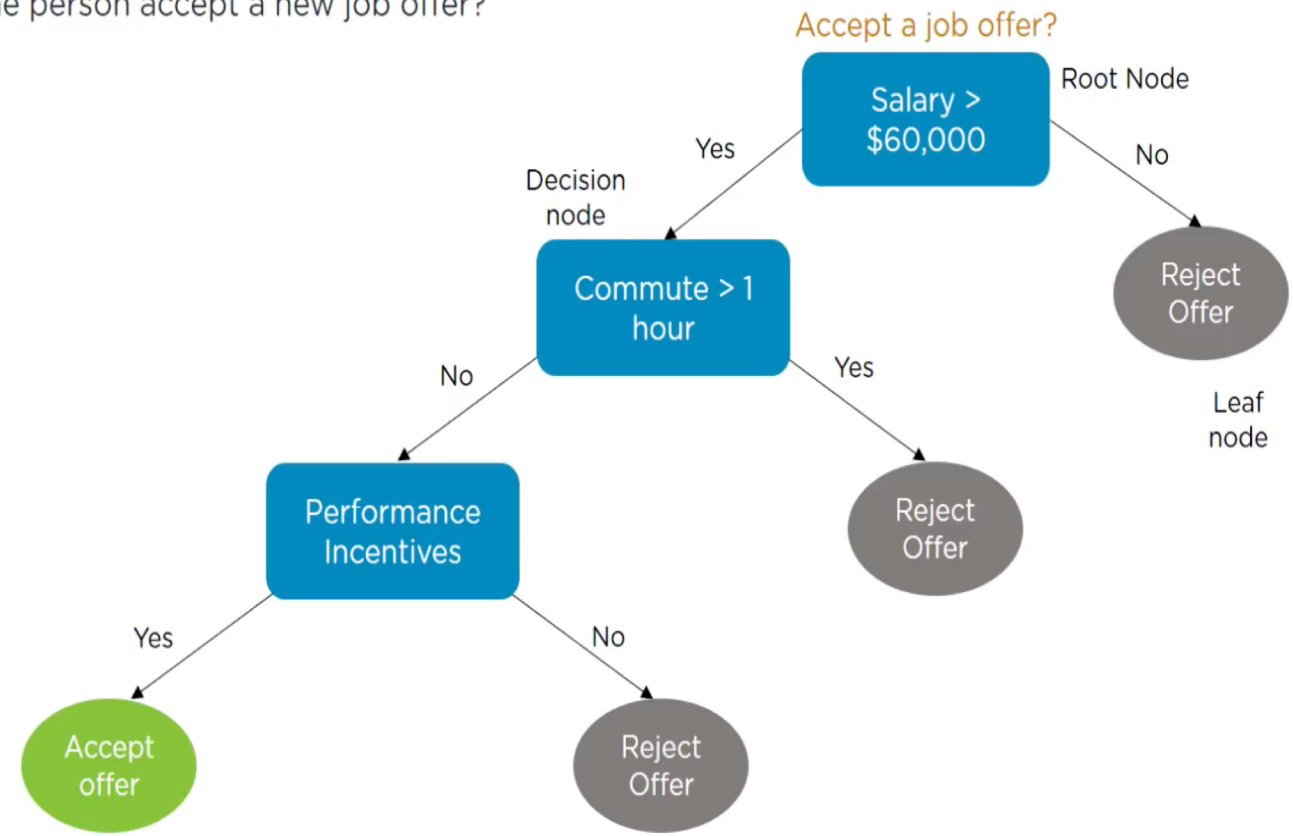
Accuracy:
(TN+TP)/N
(79+38)/134 = 0.87

Misclassification Rate:
(FP+FN)/N
(6+11)/134 = 0.13

Decision Tree

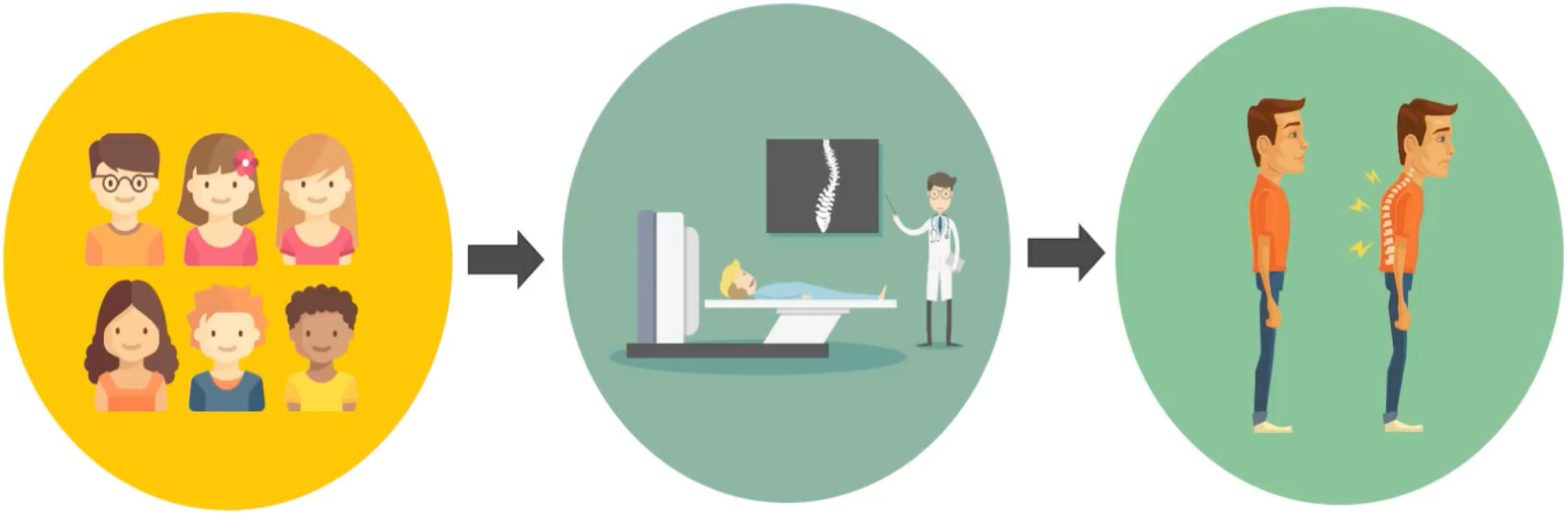
Decision Tree is a tree where each node represents a feature (attribute), each link (branch) represents a decision and each leaf represents an outcome (categorical or continuous value).

Should the person accept a new job offer?



Implementation of Decision Tree

Decision Tree and Random Forest: Does Kyphosis exist after a surgery?



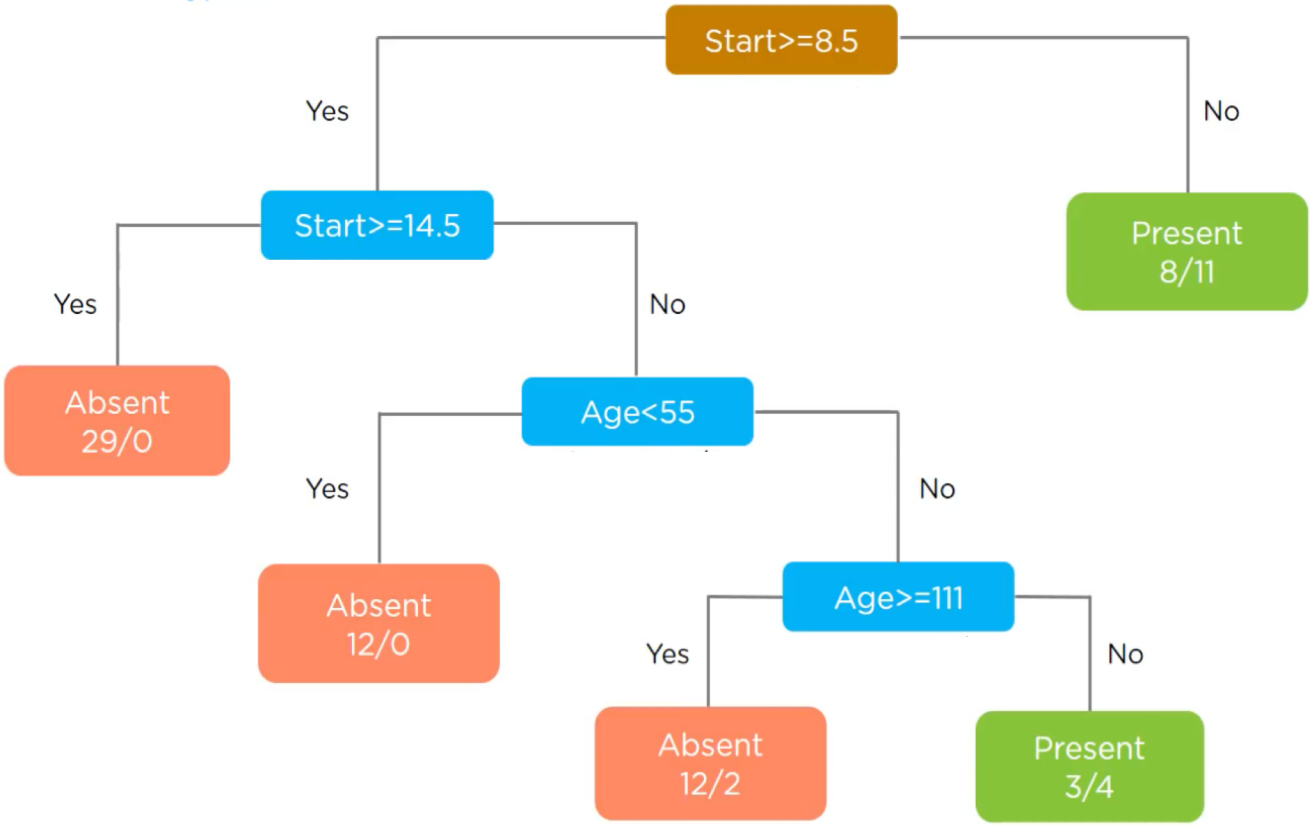
Bunch of kids

Kyphosis surgery

Predict kyphosis present or absent?

Implementation of Decision Tree

Classification Tree for Kyphosis:



Implementation of Decision Tree

4. Split the data into Train and Test set:

Spliting the dataset into training and testing 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=100)
```

5. Train a Decision Tree:

Training a Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
dtree = DecisionTreeClassifier()
```

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

Implementation of Decision Tree

6. Predict the Model:

Predicting the Model

```
predictions = dtree.predict(X_test)
predictions
```

```
array(['absent', 'present', 'present', 'absent', 'present', 'absent',
       'absent', 'absent', 'present', 'absent', 'absent', 'absent',
       'absent', 'absent', 'present', 'present', 'absent', 'absent',
       'absent', 'present', 'absent', 'absent', 'absent', 'absent',
       'present'], dtype=object)
```

Implementation of Decision Tree

7. Evaluate the Model:

Evaluation of the Model

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

     absent       0.88      0.64      0.74        22
    present       0.11      0.33      0.17         3

avg / total       0.78      0.60      0.67        25
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[15  7]
 [ 2  1]]
```

Understanding the confusion matrix:

N = 25	Predicted: No	Predicted: Yes
Actual: No	TN=15	FP=7
Actual: Yes	FN=2	TP=1

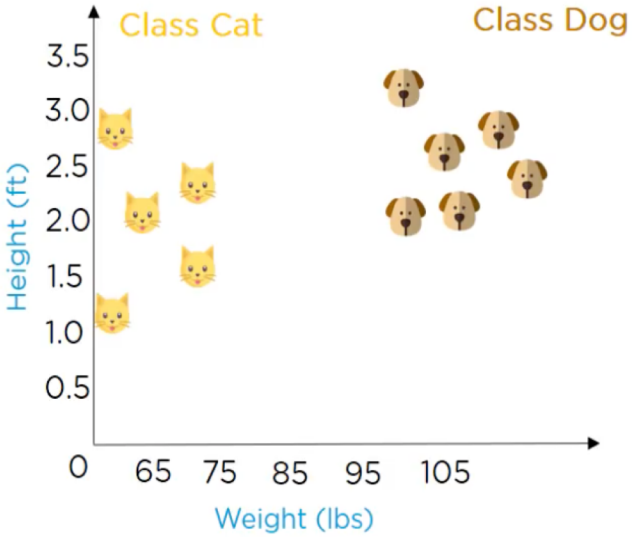
Accuracy:
(TN+TP)/N
(15+1)/25 = 0.64

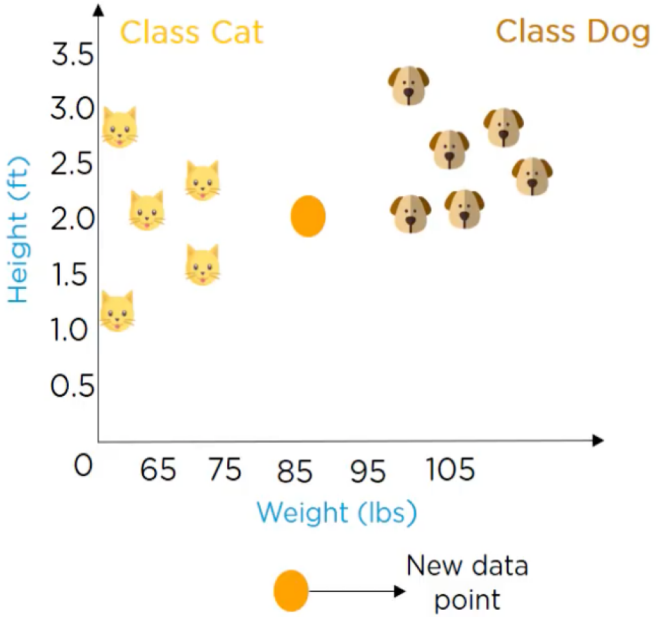
Misclassification Rate:
(FP+FN)/N
(7+2)/25 = 0.36

K Nearest Neighbors (KNN)

K Nearest Neighbors: KNN is a Classification algorithm generally used to predict categorical values.

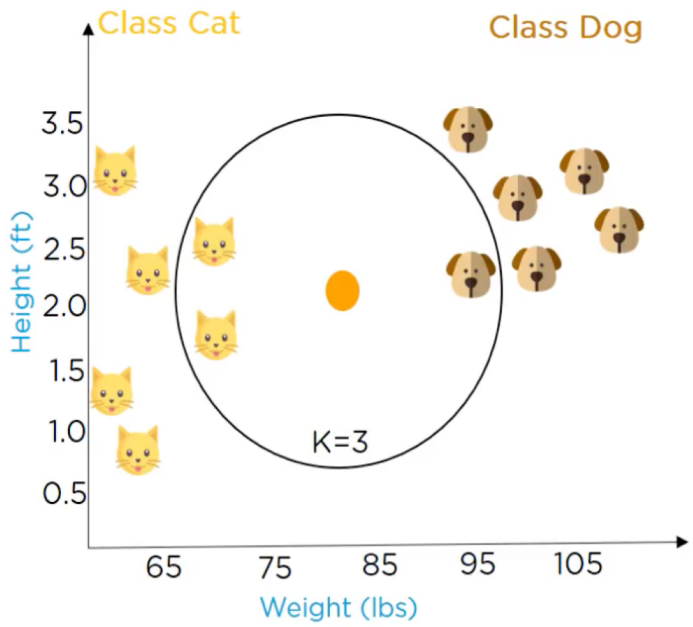
To find if a new data point is a  or a  ?





K Nearest Neighbors

Choosing a K will define what class a new data point is assigned to:



If K=3, the new data point belongs to class Cat

Implementation of KNN

6. Fit KNN to Train set:

Fitting KNN to the Training dataset

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

7. Predict the Test set results:

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
```

REFERENCE

- https://scikit-learn.org

scikit-learn

Machine Learning in Python

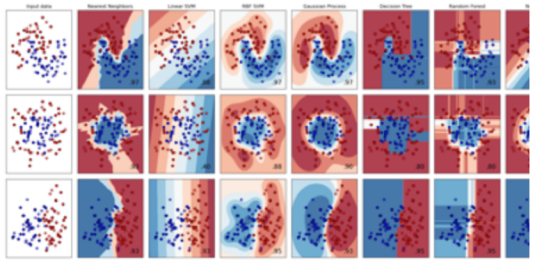
Getting Started | Release Highlights for 0.24 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

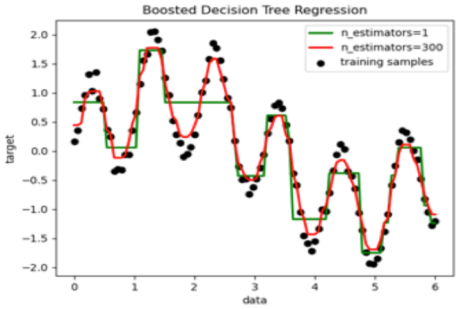


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

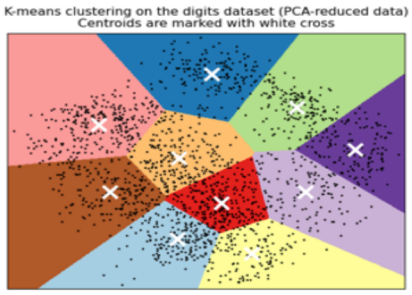


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

PYTHON FOR DATA SCIENCE CHEAT SHEET

Python Scikit-Learn

Introduction

Scikit-learn: "sklearn" is a machine learning library for the Python programming language. Simple and efficient tool for data mining, Data analysis and Machine Learning.

Importing Convention - import sklearn

Preprocessing

Data Loading

- **Using NumPy:**

```
>>>import numpy as np
>>>a=np.array([(1,2,3,4),(7,8,9,10)],dtype=int)
>>>data = np.loadtxt('file_name.csv',
   delimiter=',')
```

- **Using Pandas:**

```
>>>import pandas as pd
>>>df=pd.read_csv('file_name.csv',header=0)
```

Train-Test Data

```
>>>from sklearn.model_selection
import train_test_split

>>> X_train, X_test, y_train, y_test =
train_test_split(X,y,random_state=0)
```

Data Preparation

- **Standardization**

```
>>>from sklearn.preprocessing import
StandardScaler
>>>get_names = df.columns
>>>scaler =
preprocessing.StandardScaler()
>>>scaled_df = scaler.fit_transform(df)
>>>scaled_df =
pd.DataFrame(scaled_df,
columns=get_names)m
```

- **Normalization**

```
>>>from sklearn.preprocessing import
   Normalizer
>>>pd.read_csv("File_name.csv")
 >>>x_array = np.array(df['Column1'])
#Normalize  Column1
>>>normalized_X =
preprocessing.normalize([x_array])
```

Working On Model

Model Choosing

Supervised Learning Estimator:

- **Linear Regression:**

```
>>> from sklearn.linear_model import
LinearRegression
>>> new_lr=
LinearRegression(normalize=True)
```

- **Support Vector Machine:**

```
>>> from sklearn.svm import SVC
>>> new_svc = SVC(kernel='linear')
```

- **Naive Bayes:**

```
>>> from sklearn.naive_bayes import
GaussianNB
>>> new_gnb = GaussianNB()
```

- **KNN:**

```
>>> from sklearn import neighbors
>>>
knn=neighbors.KNeighborsClassifier(n_ne
ighbors=1)
```

Unsupervised Learning Estimator:

- **Principal Component Analysis (PCA):**

```
>>> from sklearn.decomposition import
PCA
>>> new_pca=PCA(n_components=0.95)
```

- **K Means:**

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=5,
random_state=0)
```

Train-Test Data

Supervised:

```
>>>new_ lr.fit(X,y)
>>> knn.fit(X_train, y_train)
>>>new_svc.fit(X_train, y_train)
```

Unsupervised :

```
>>> k_means.fit(X_train)
>>> pca_model_fit =
new_pca.fit_transform(X_train)
```

Post-Processing

Prediction

Supervised:

```
>>> y_predict =
new_svc.predict(np.random.random((3,5)))
>>> y_predict = new_lr.predict(X_test)
>>> y_predict = knn.predict_proba(X_test)
```

Unsupervised:

```
>>> y_pred = k_means.predict(X_test)
```

Model Tuning

Grid Search:

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3), "metric":
        ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization:

```
>>> from sklearn.grid_search import Randor
>>> params = {"n_neighbors": range(1,5), "w
["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimat
param_distributions=params, cv=4, n_iter=8
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

Evaluate Performance

Classification:

1. **Confusion Matrix:**

```
>>> from sklearn.metrics import
      confusion_matrix
>>> print(confusion_matrix(y_test,
      y_pred))
```

2. **Accuracy Score:**

```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import
      accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Regression:

1. **Mean Absolute Error:**

```
>>> from sklearn.metrics import mean_absolute_error

>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_predict)
```

2. **Mean Squared Error:**

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_predict)
```

3. **R^2 Score :**

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_predict)
```

Clustering:

1. **Homogeneity:**

```
>>> from sklearn.metrics import
homogeneity_score
>>> homogeneity_score(y_true,
y_predict)
```

2. **V-measure:**

```
>>> from sklearn.metrics import
v_measure_score
>>> metrics.v_measure_score(y_true,
y_predict)
```

Cross-validation:

```
>>> from
sklearn.cross_validation
import cross_val_score
>>>
print(cross_val_score(knn,
X_train, y_train, cv=4))
>>>
print(cross_val_score(new_
lr, X, y, cv=2))
```

FURTHERMORE:
Python for Data Science Certification Training Course

IntelliPaat

THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大学