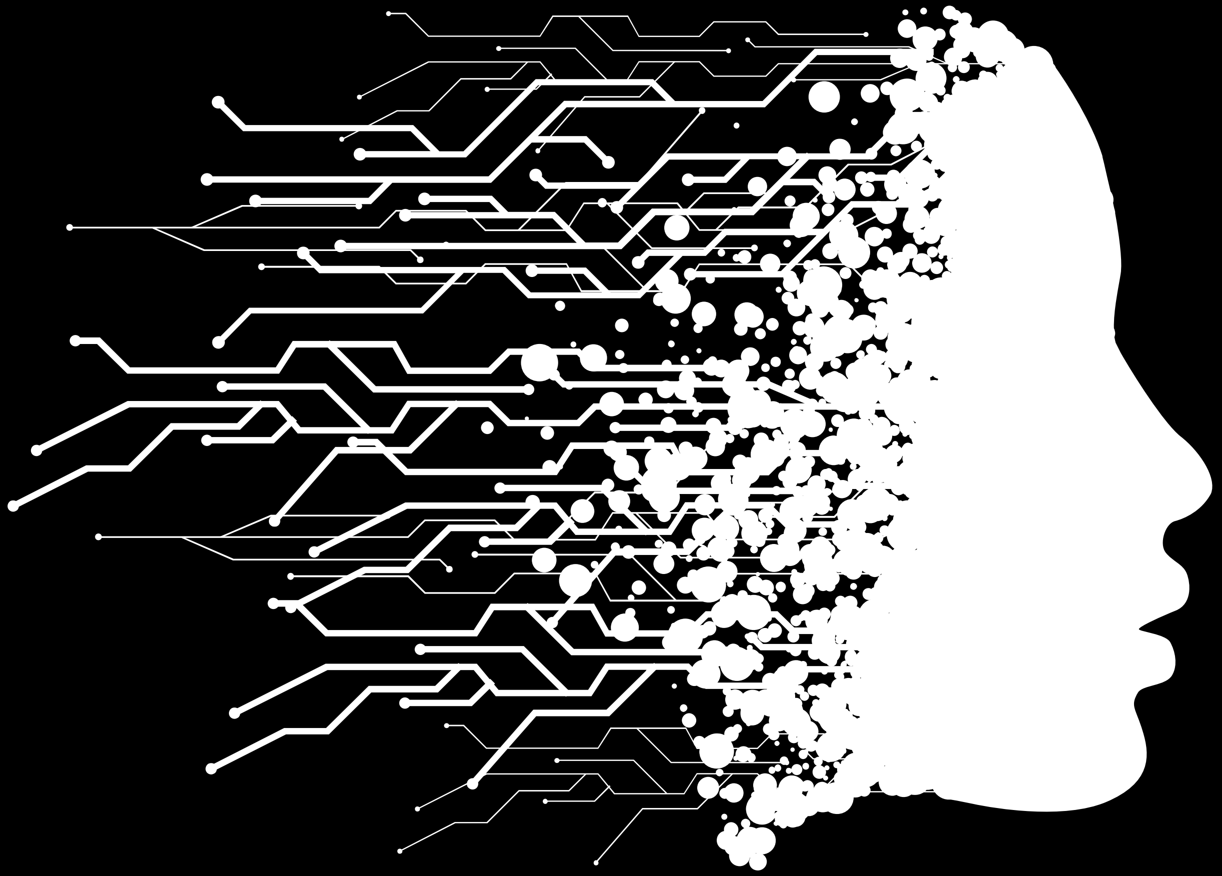


Machine Learning With Spark (BD127)

We will be starting soon

Day 2

Welcome to Day 2 Spark Machine Learning

Please perform PRE-WORK

1. Access the virtual Lab using link https://html.inspiredvlabs.com Use the username TEKMD190-XX (replace XX with your number) and password **TEKBD127!23**

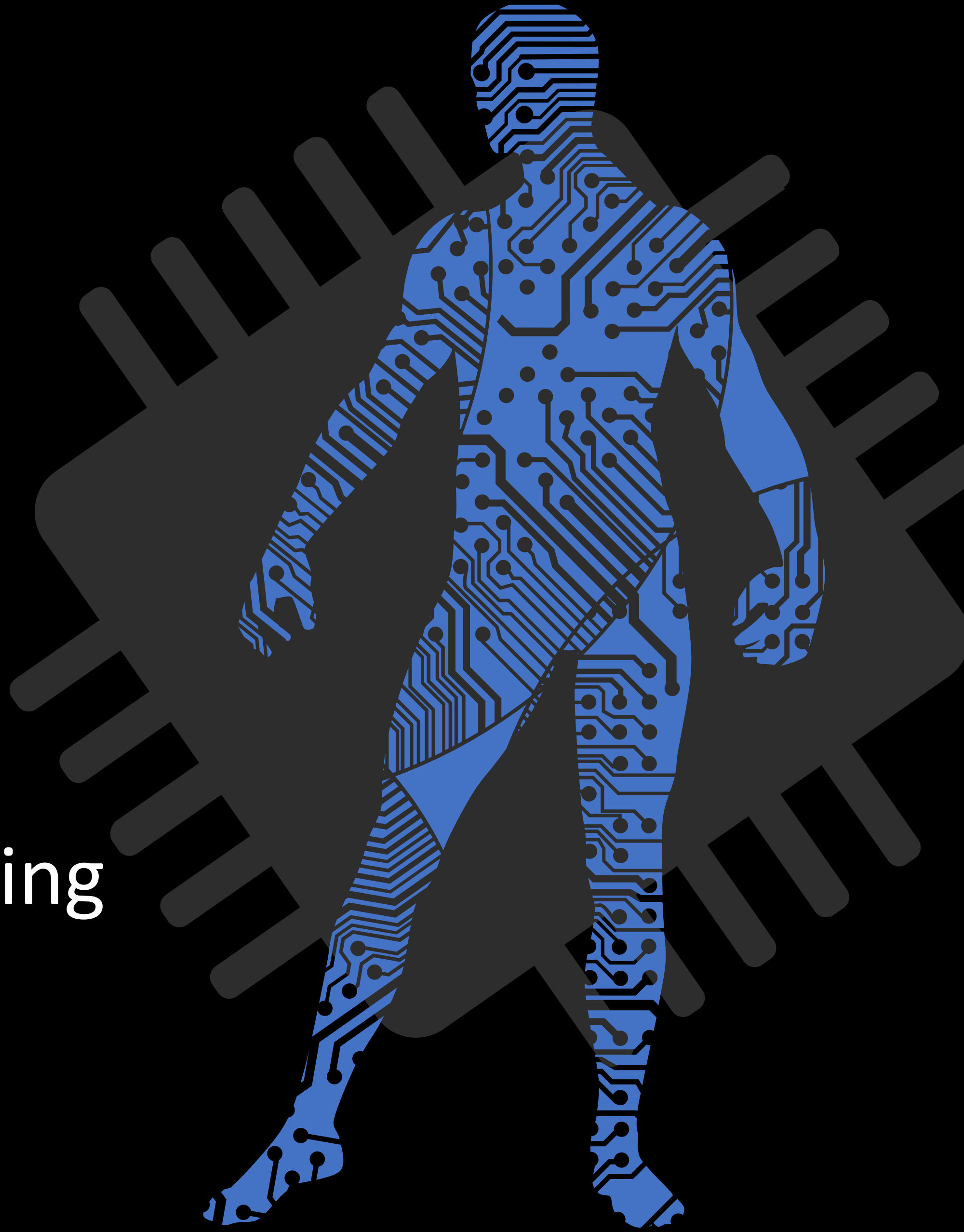
https://tinyurl.com/bdtSparkML

Last Name	First Name	Login Id
ALCEDO MORENO	ALVARO	TEKMD190-01
BOGADAPATI	BHAVANI	TEKMD190-02
BOOSTANI	ANOUSH	TEKMD190-03
FRENCH	CHRIS	TEKMD190-04
FRINO	MASSIMILIANO	TEKMD190-05
KULKARNI	ALPESH	TEKMD190-06
MA	CUONG	TEKMD190-07
MADAGANI	SRINIVASA	TEKMD190-08
MATULIS	STEPHEN	TEKMD190-09
MIAO	HUALING	TEKMD190-10
MILLER	KENT	TEKMD190-11
OBRIEN	CHARLES	TEKMD190-12
SELVARAJ	RAJESH KHANNA	TEKMD190-13
VINCENT	SWAROOP	TEKMD190-14
YOUSSEF	MINA	TEKMD190-15

We will be starting soon

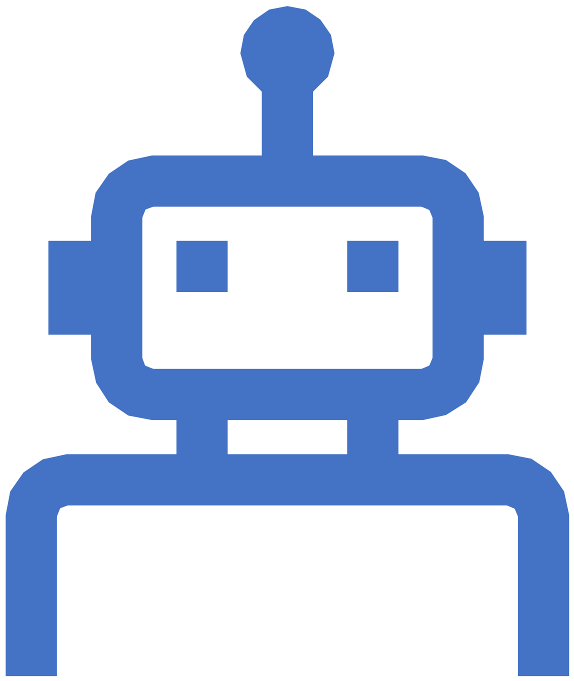
Agenda – Day 2

1. Recap of Day 1
2. Use Case Real Estate (Homework)
3. Project Initiation
4. Spark ML Processing
5. Feature Engineering and Data Cleaning
6. More Algorithms
7. Model Evaluations and Metrics



Recap – Day 1

1. What is Machine Learning?
2. Spark Overview (Ecosystems before and after)
3. Spark ML Development
4. Machine Learning Techniques (CCRA)
    a. Supervised
    b. Unsupervised
5. Machine Learning Development (DIAPERS)
6. Data (Clean, Coverage, Complete)
7. Statistics Brush up
8. Popular Algorithms
9. Linear Regression
10. Multiple hands-on exercises



Assignment

- Open notebook – “SparkLab/Spark Linear Regression Real Estate”
- Loads data from sci-kit learn (sklearn) dataset – “load_boston”
- Data exploration done with pandas and the data frame is converted to Spark data frame
- Write code for Linear Regression (Marked in Red)

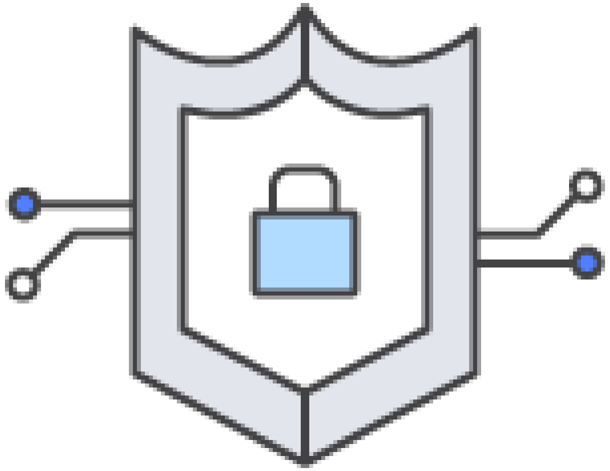
Use case: Boston Real Estate Data

- Dataset – from scikit-learn datasets
- Linear Regression example with following features:

CRI: Per capita crime rate by town
ZN: Proportion of residential land zoned for lots over 25,000 sq. ft
INDUS: Proportion of non-retail business acres per town
CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX: Nitric oxide concentration (parts per 10 million)
RM: Average number of rooms per dwelling
AGE: Proportion of owner-occupied units built prior to 1940
DIS: Weighted distances to five Boston employment centers
RAD: Index of accessibility to radial highways
TAX: Full-value property tax rate per $10,000
PTRATIO: Pupil-teacher ratio by town
B: $1000(Bk — 0.63)^2$, where Bk is the proportion of [people of African American descent] by town
LSTAT: Percentage of lower status of the population
PRICE: Median value of owner-occupied homes in $1000s

Spark Lab/Spark Linear Regression Real Estate

Feature Engineering and Data Cleaning



Data Cleaning Demo

- Open file ‘SparkExamples/Data Cleaning Using Spark’ using Jupyter
- Look for missing values
- Binary categorical data replacement e.g. Male, Female
- N-nary categorical data replacement e.g. Embarked
- Using Imputer to replace missing age values
- Drop an entire sample due to many missing features

Titanic Dataset

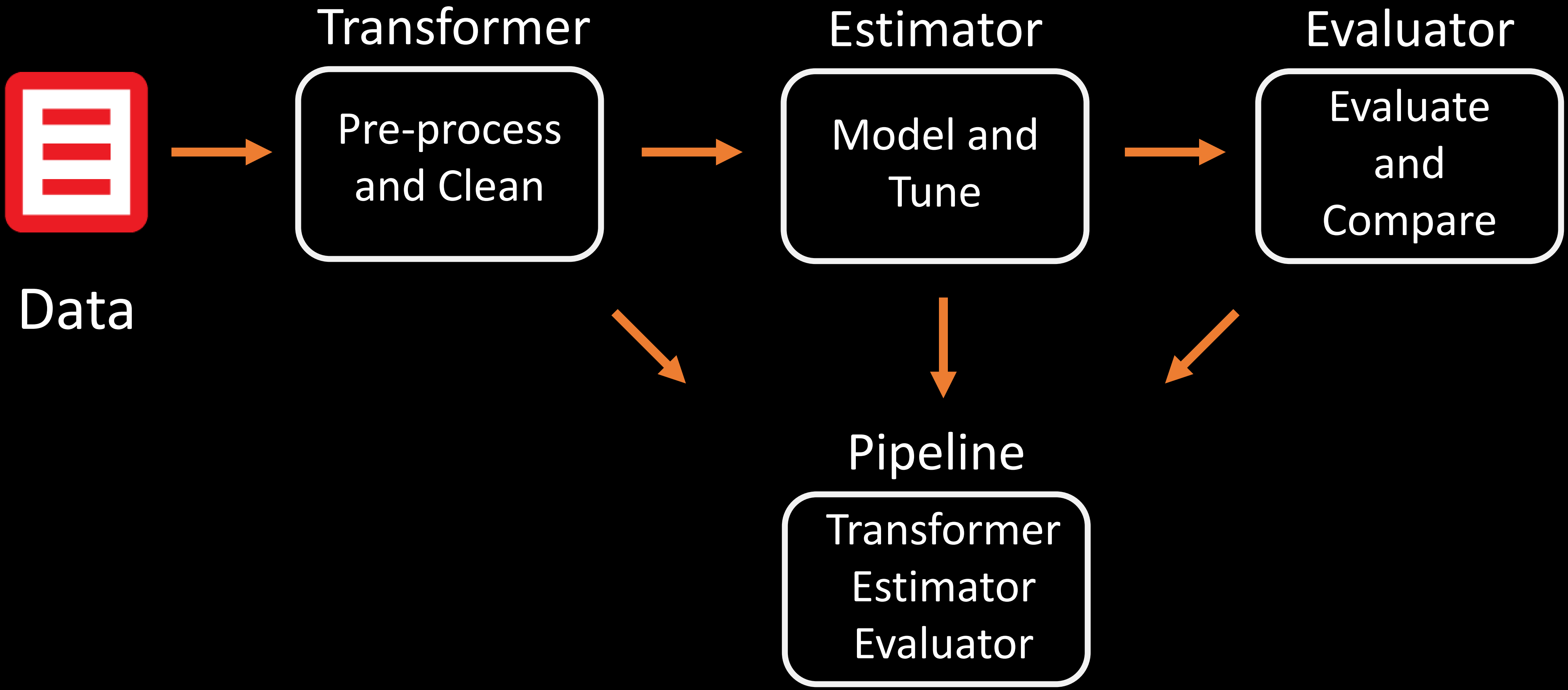
- Will use titanic data set to predict whether a passenger survived or not
- Data consists of following:
  - Pclass - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
  - sex – Gender
  - sibsp - Number of siblings/Spouses
  - parch - Number of parents/children
  - fare - travel fare
  - Embarked - (C = Cherbourg; Q = Queenstown; S = Southampton)
  - boat - Life boat
  - body - Body identification number
  - home.dest – Destination
  - ticket - Ticket number
  - cabin - Cabin number
  - name - Passenger name
  - survived - (0: No, 1: Yes)

Spark ML Processing

Spark ML

Data

Spark ML


Transformer
Estimator
Evaluator
Pre-process and Clean
Model and Tune
Evaluate and Compare
Data
Pipeline
Transformer
Estimator
Evaluator

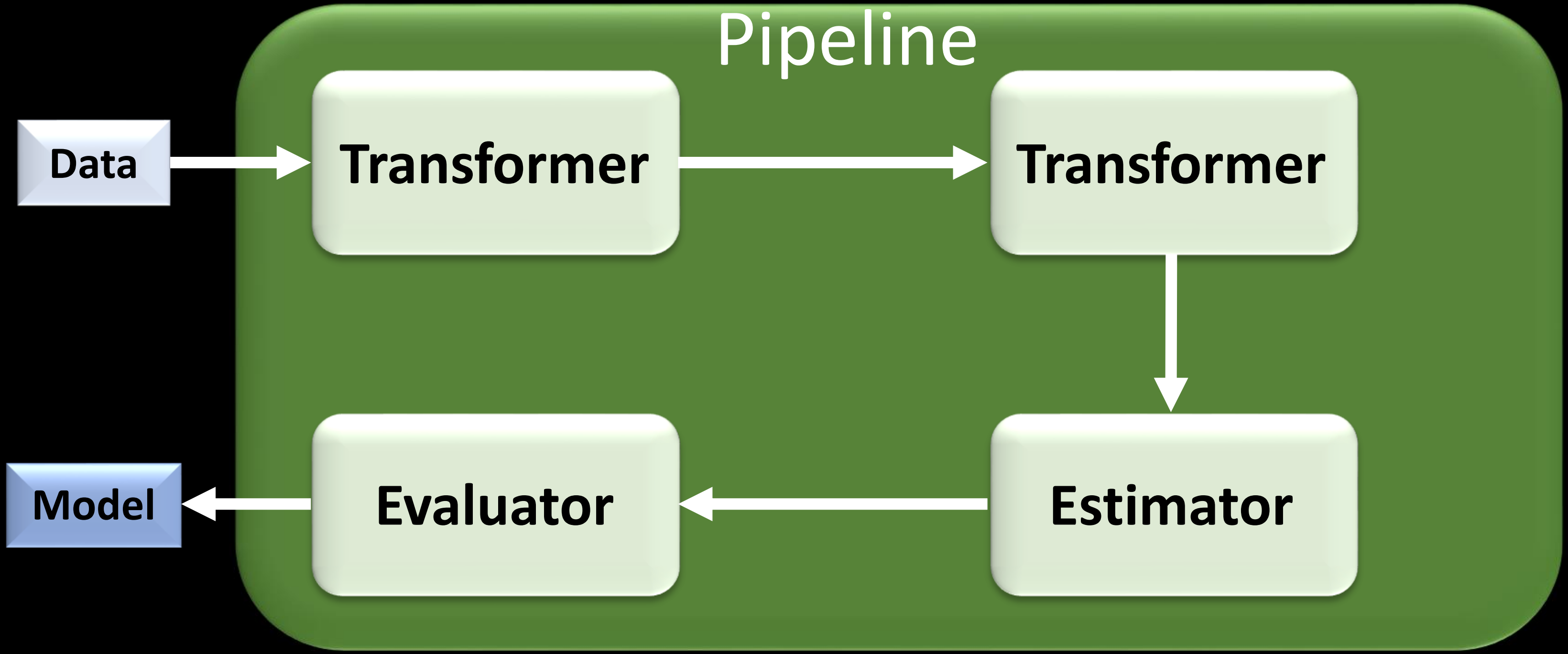
Evaluator Types

Evaluator	Description
BinaryClassifierEvaluator	Binary Classification model evaluator
MultiClassClassificationEvaluator	Multiple Class Classification model evaluator
RegressionEvaluator	Regression model evaluator
ClusteringEvaluator	Clustering model evaluator

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# Evaluate model
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")
evaluator.evaluate(predictions)
```

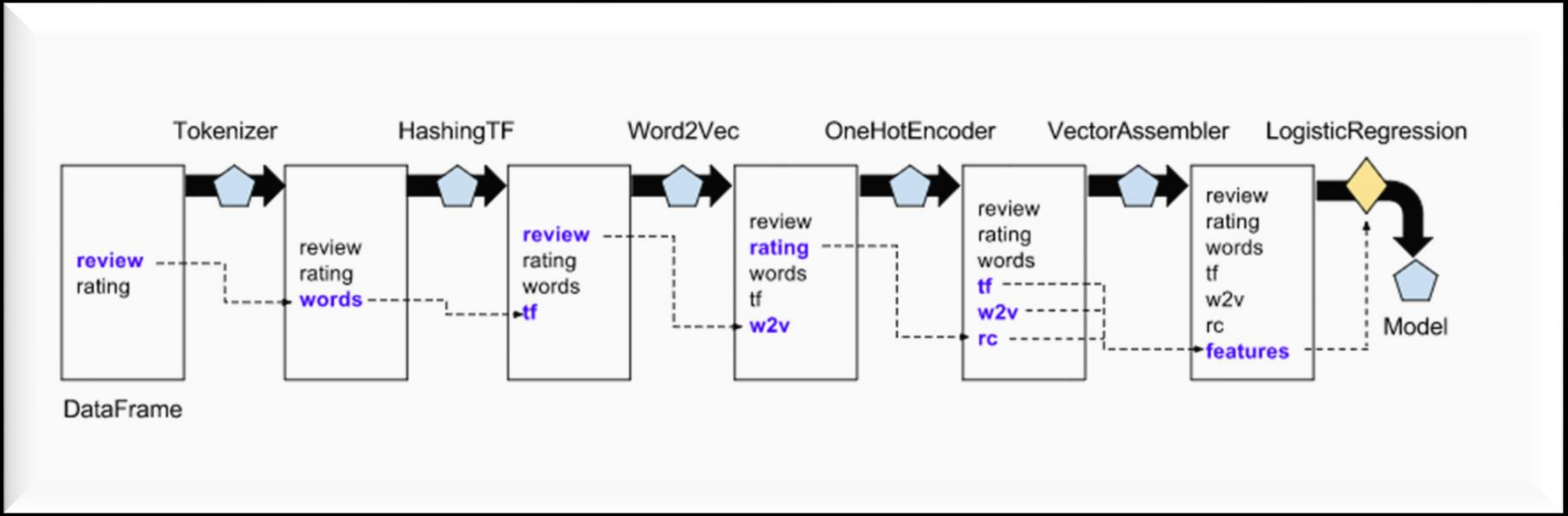
Pipeline



- A machine learning work-flow
- Made up of number of stages
- Can be persisted

Pipeline Example

```
# Configure pipeline stages
tok         = Tokenizer(inputCol="review", outputCol="words")
htf         = HashingTF(inputCol="words", outputCol="tf", numFeatures=200)
w2v         = Word2Vec(inputCol="review", outputCol="w2v")
ohe         = OneHotEncoder(inputCol="rating", outputCol="rc")
va          = VectorAssembler(inputCols=["tf", "w2v", "rc"], outputCol="features")
lr          = LogisticRegression(maxIter=10, regParam=0.01) # Build the pipeline
pipeline    = Pipeline(stages=[tok, htf, w2v, ohe, va, lr]) # Fit the pipeline
model       = pipeline.fit(train_df)
```



Project Initiation
Income Prediction

Spark Income Prediction

- Objective is to predict if a person’s income will be <= 50K or > 50K using number of features
- Code examples for Transformer, Estimator, Evaluator and Pipeline
- Will implement different algorithms with this dataset

Project - Dataframe

Age	Sex	Race	Income
39	Male	White	<=50K
50	Female	Black	>50K
38	Female	Asian	<=50K
53	Male	Other	>50K

Project - StringIndexer

Age	Sex	Race	Income
39	0	0	0
50	1	1	1
38	1	2	0
53	0	3	1

Project – One-Hot Encoding Estimator

Age	Sex	Race	Race_ White	Race_ Black	Race_ Asian	Race_ Other	Income
39	0	0	1	0	0	0	0
50	1	1	0	1	0	0	1
38	1	2	0	0	1	0	0
53	0	3	0	0	0	1	1

Project – Sparse Vector

Age	Sex	Race	Race_ White	Race_ Black	Race_ Asian	Race_ Other	Income
39	0	0	1	0	0	0	0
50	1	1	0	1	0	0	1
38	1	2	0	0	1	0	0
53	0	3	0	0	0	1	1

Sparse Vector Encode: [0, 8, [0,3], [39, 1]]

Age	Sex	Race	Race_ White	Race_ Black	Race_ Asian	Race_ Other	Income
39	0	0	1	0	0	0	0
50	1	1	0	1	0	0	1
38	1	2	0	0	1	0	0
53	0	3	0	0	0	1	1

Sparse Vector Encode: [1, 8, [0,1,4,7], [50,1,1,1]]

Age	Sex	Race	Race_ White	Race_ Black	Race_ Asian	Race_ Other	Income
39	0	0	1	0	0	0	0
50	1	1	0	1	0	0	1
38	1	2	0	0	1	0	0
53	0	3	0	0	0	1	1

Sparse Vector Assembler:

Age	Sex	Race	Race_ White	Race_ Black	Race_ Asian	Race_ Other	Income
39	0	0	1	0	0	0	0
50	1	1	0	1	0	0	1
38	1	2	0	0	1	0	0
53	0	3	0	0	0	1	1

[0, 8, [0,3], [39,1] ……………[3,8], [0,6,7], [53, 1, 1]]

Spark Machine Learning

Training

model.fit(dataset)

Evaluate with Test Data

model.transform(test_dataset)

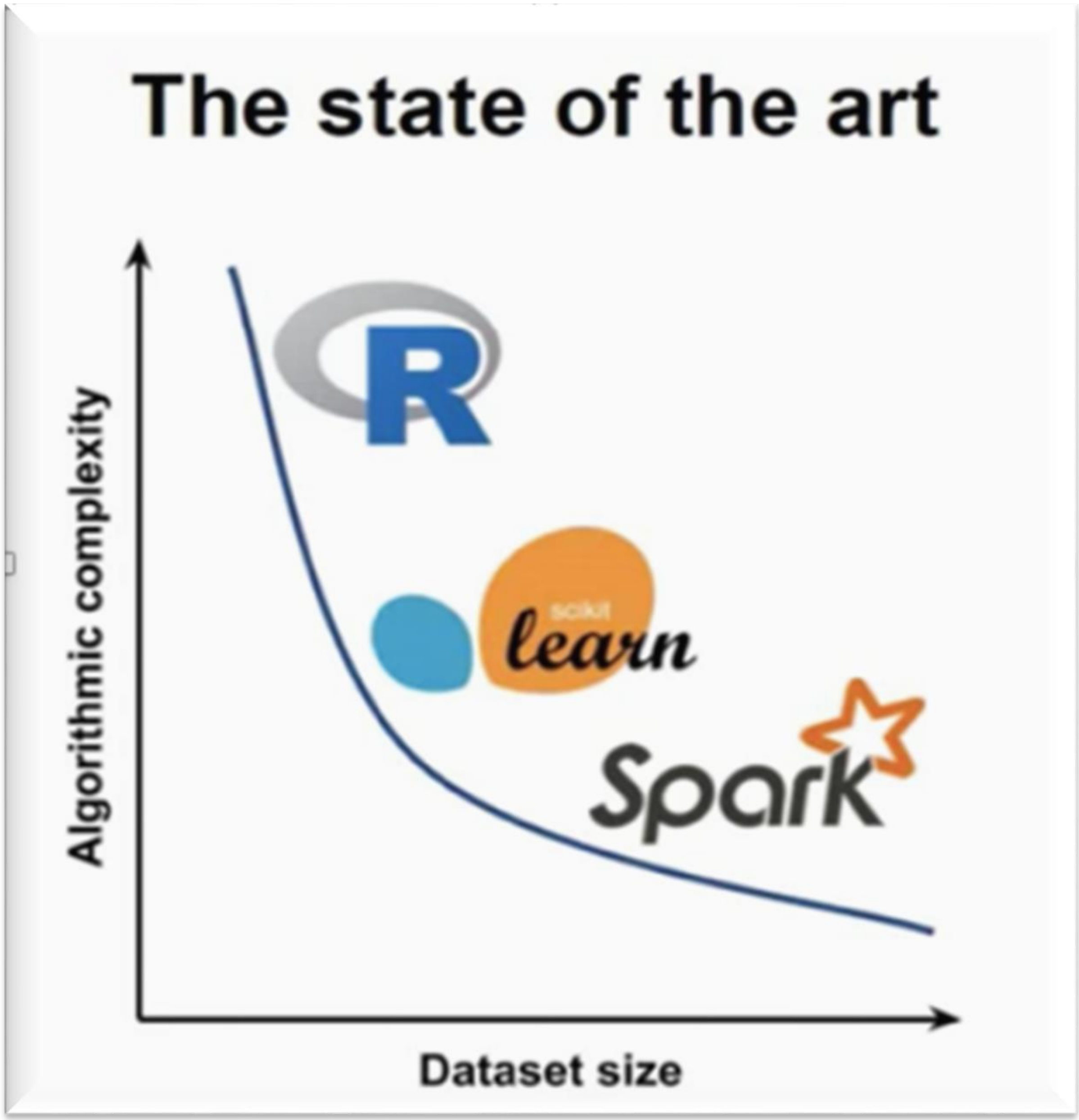
Evaluation the Model

model.evaluate(predictions, test_dataset)

Spark and Sci-kit Learn

Function	Spark ML	Scikit-Learn
Small and Medium datasets (in megabytes)	Spark will perform better (depends on your deployment environment)	For larger datasets scikit-learn will require lot of RAM
Distributed Deployment	Spark is designed for it	Need distributed support
Model building	Less flexible – does not have lot of APIs. You will have to write more code	More flexible and efficient
Visualization	Cumbersome to implement	Hands-down more elegant support

Spark v/s Other ML Implementations



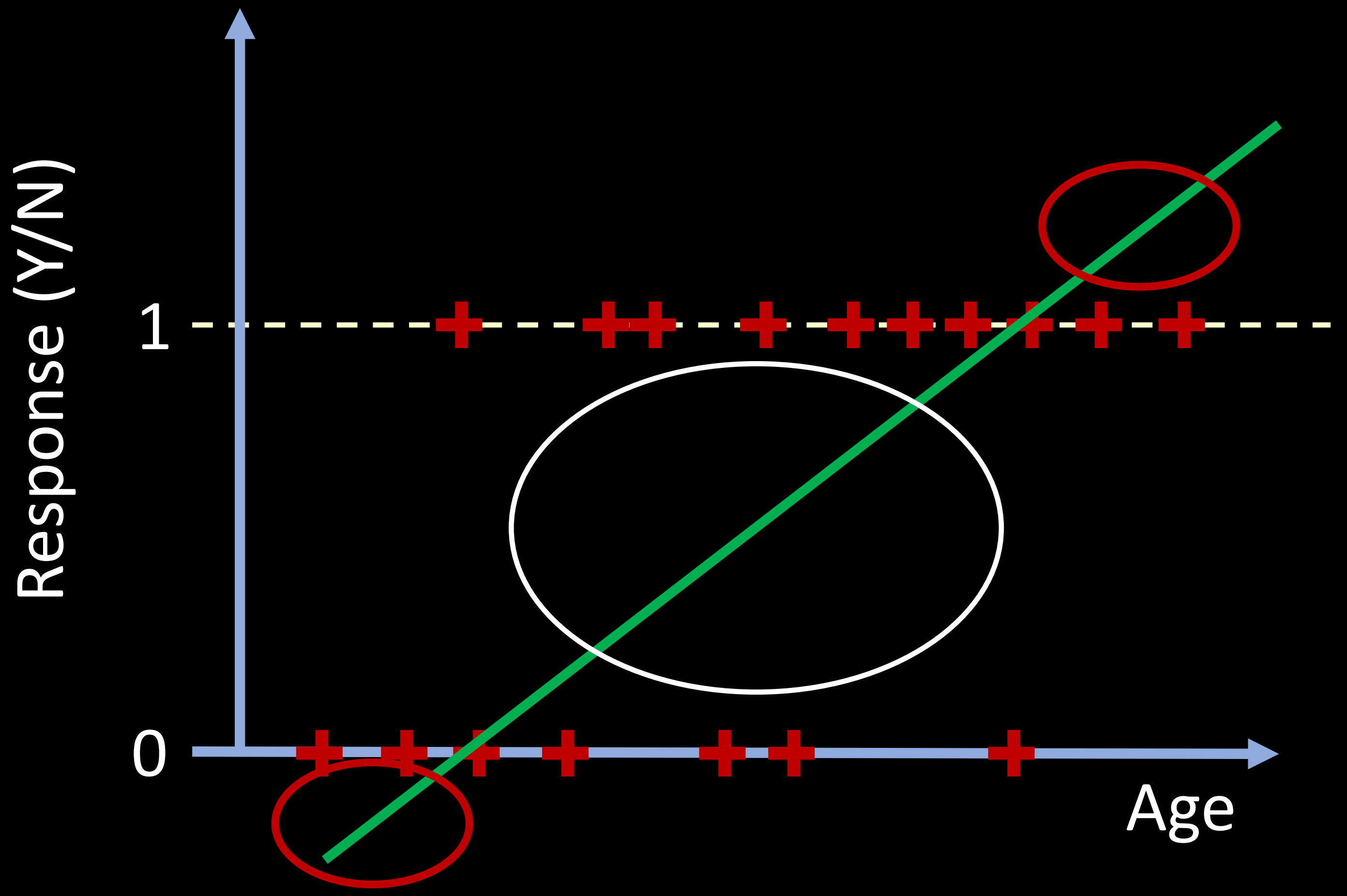
Logistic Regression

Example: Logistic Regression

Social Network Ads

User ID	Gender	Age	Estimated Salary	Purchased
15624510	Male	19	19000	Yes
15810944	Male	35	20000	No
15668575	Female	26	43000	No
15603246	Female	27	57000	Yes
15804002	Male	19	76000	No
15728773	Male	27	58000	No
15598044	Female	27	84000	No
15694829	Female	32	150000	Yes
15600575	Male	25	33000	No
15727311	Female	35	65000	No
15570769	Female	26	80000	No
15606274	Female	26	52000	No

Linear Regression Challenges



Logistic Regression

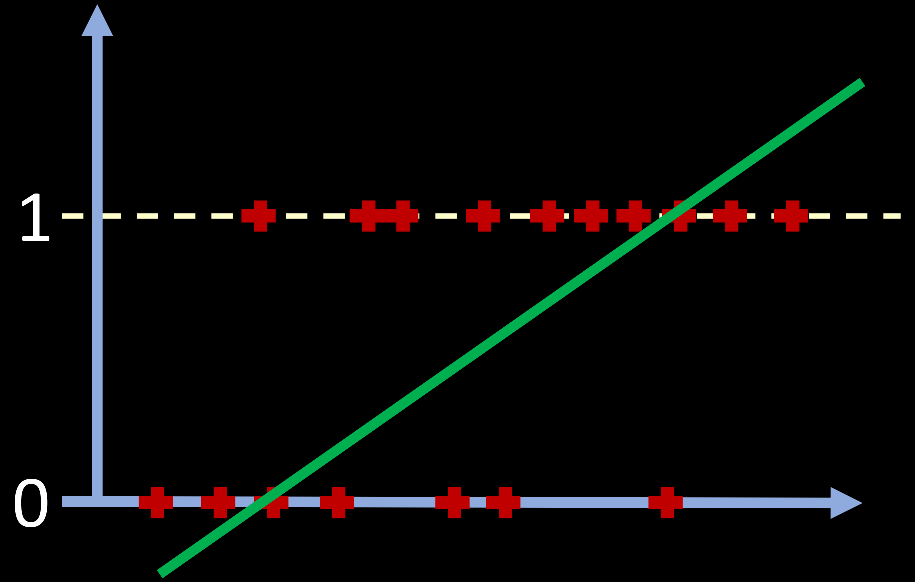
$$y = b_0 + b_1 * x_1$$

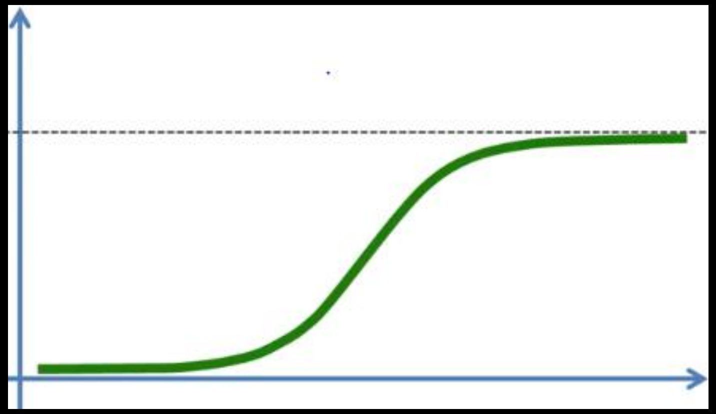
Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$$

This is the formula for logistic regression

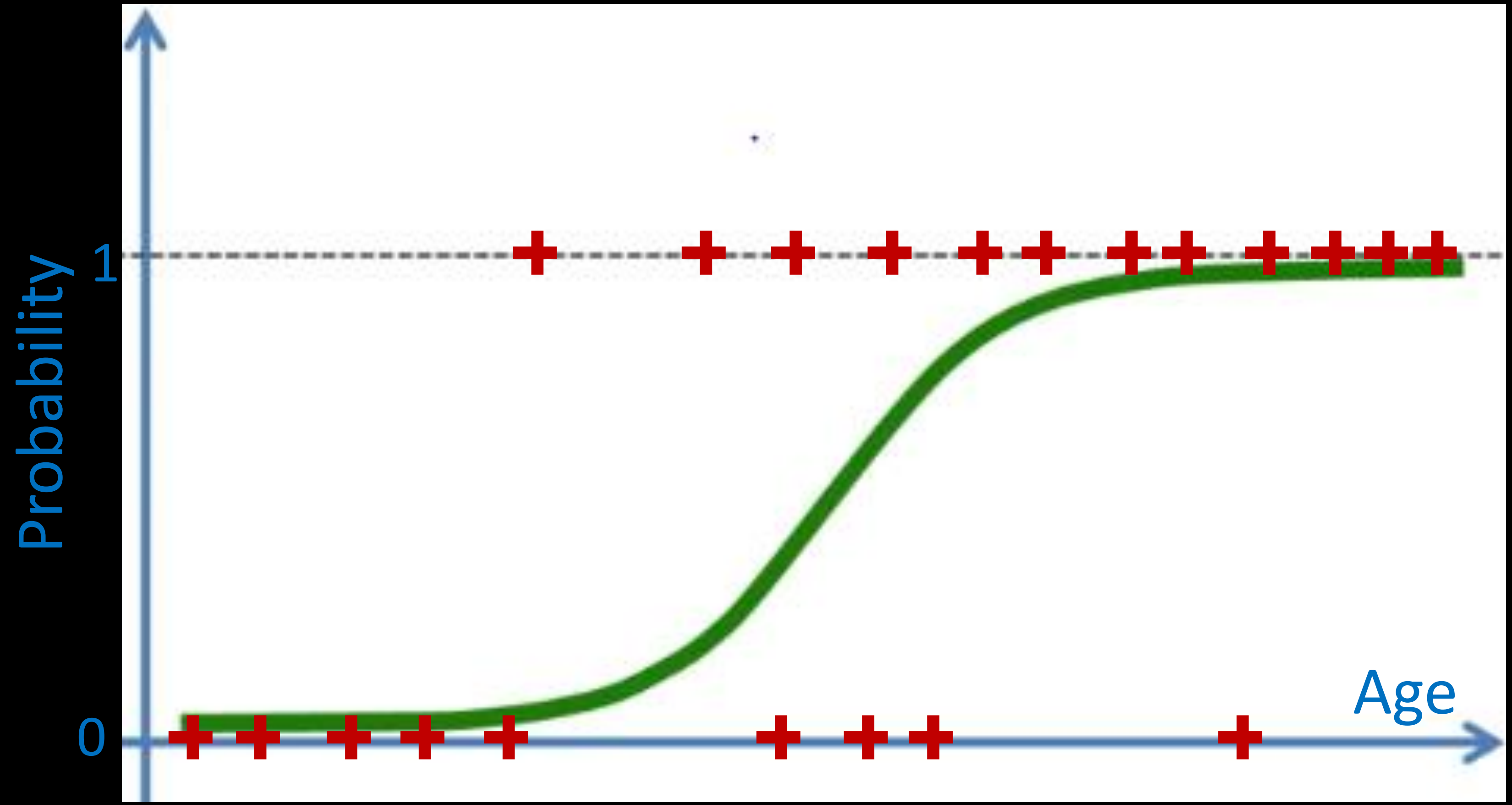




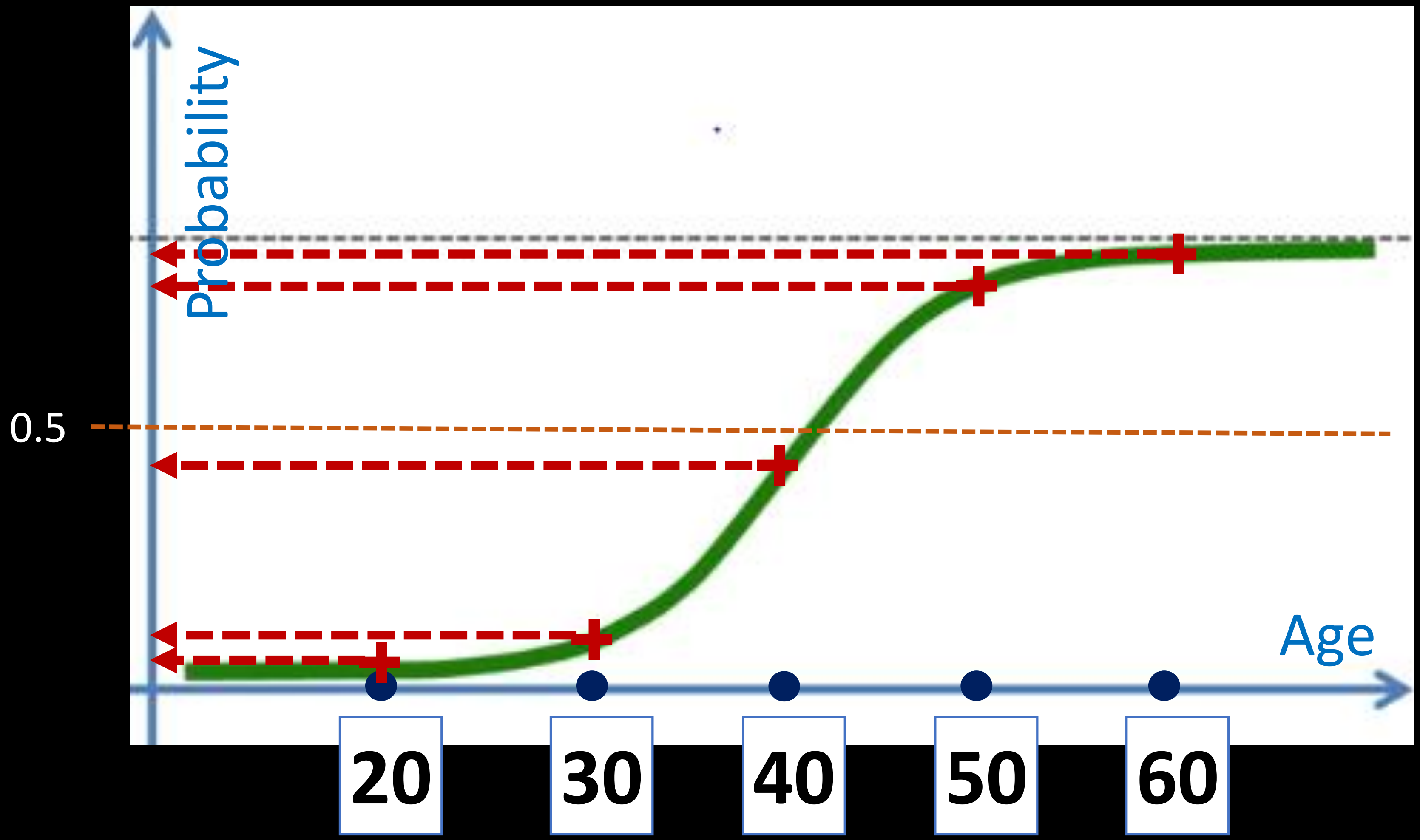
Logistic Regression

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$$

Age = Independent Variable (X)
Probability = Dependent Variable (y)



Logistic Regression



positive class = 1
negative class = 0

If probability >= 0.5: predict 1
If probability < 0.5: predict 0

Linear v/s Logistic Regression

Criteria	Linear Regression	Logistic Regression
Basic Definition	Data is modelled as a straight line	Is used to model certain probability of event/class happening e.g. pass/fail, yes/no, etc.
Linear relation between independent variable and dependent variable	Required	Not Required
Independent variables	Can be correlated to with each other (mostly in multiple regression)	Should not be correlated with each other
Predictions	Values can be any number	Values between 0 and 1

Model Metrics

Model Performance Metrics

- **False Positive**
  - Model predicted a positive outcome, but it was negative. We predicted an event that did not occur
  - It is like a warning sign e.g. the prediction was that an earthquake will occur, but it did not occur
- **False Negative**
  - Model predicted that there won’t be an event, but the event occurred
- Also have True Positive and True Negative

Confusion Matrix

		Predictions	
		0	1
Actual	0	**True Negative**	**False Positive**
	1	**False Negative**	**True Positive**

Confusion Matrix - Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Classification Report

	precision	recall	f1-score	support
0	0.85	0.94	0.89	64
1	0.86	0.69	0.77	36

$$Precision = \frac{TP}{TP + FP}$$

When it predicts a positive result, how often is it correct?
Goal is to limit number of false positive (FP)

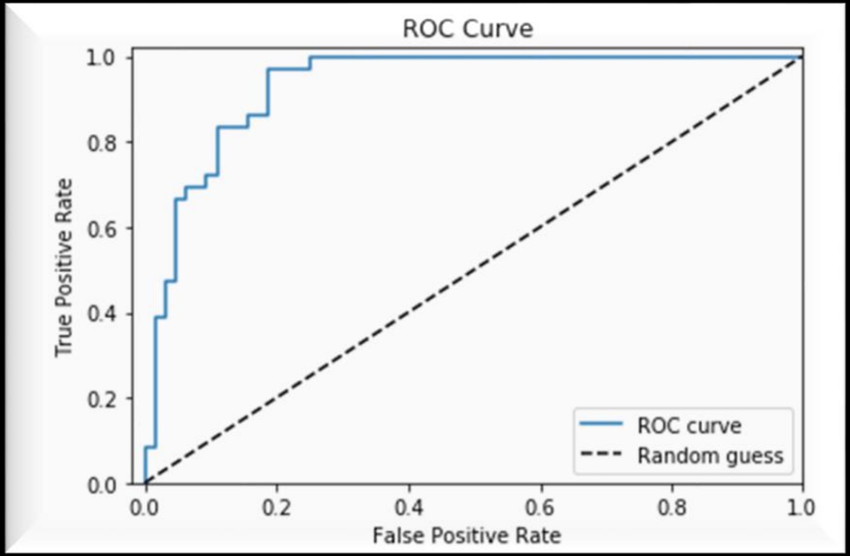
$$Recall = \frac{TP}{TP + FN}$$

When it actually predicts the positive result, how often does it predict correctly?
Goal is to limit number of false negatives (FN)

$$f1score = 2 \times \frac{precision \times recall}{precision + recall}$$

It is harmonic mean of precision and recall. Useful when we need to take both precision and recall into account

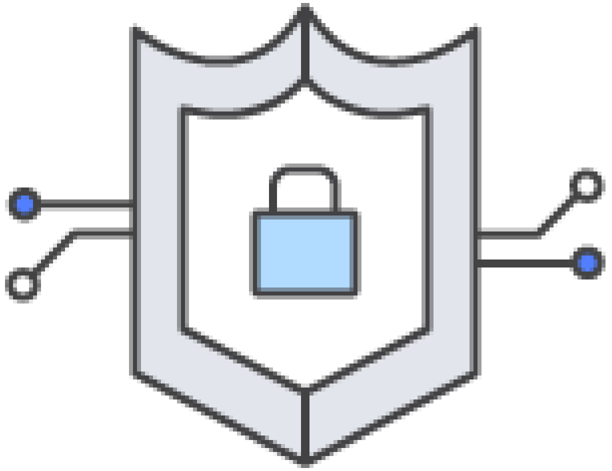
ROC Curve



ROC Curve - Receiver Operating Characteristic (ROC) curve - it is a probability curve

- A visual way to measure performance of **binary classifiers**
- When it is actually the negative result, how often does the model predict incorrectly?
- Want the curve to be as far away from the random guess line
- Area under the curve (AUC) represents the degree of separation
- Higher the AUC, the model is better at predicting 0 as 0s and 1 as 1s

Logistic Regression Demo



- Open file ‘SparkExamples/Logistic Regression Spark’ using Jupyter
- Use the Social Networking Ads data
- Apply Logistic Regression on the data
- Make predictions
- Explore training dataset metrics
- Apply binary classification evaluator

Project Code Walkthrough

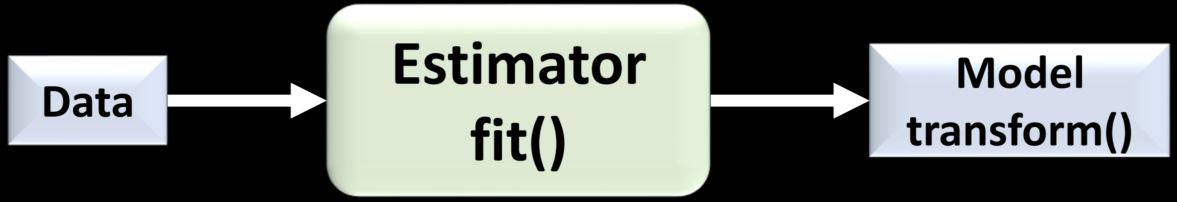
Spark Income Prediction

- Objective is to predict if a person’s income will be >= 50K or < 50K using number of features
- Code examples for Transformer, Estimator, Evaluator and Pipeline
- Will implement different algorithms with this dataset

Transformer

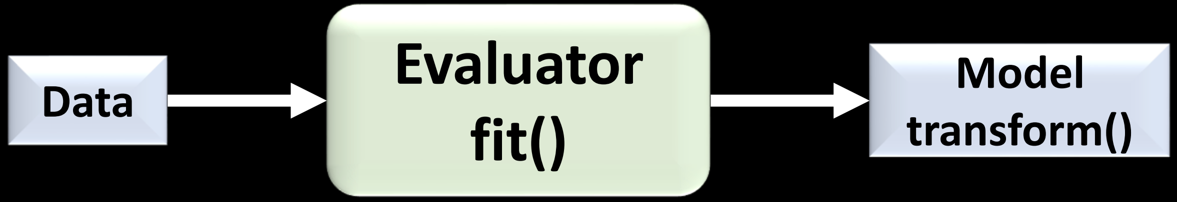
- A Transformer is an algorithm that is used to transform one data frame into another data frame
- Feature Extraction (initial processing)
- Transform data into format required by ML Algorithms
- Take input column and transform it into output column
- For example:
  - Sentences into words – Tokenizer
  - Convert categorical data into numbers e.g. Male = 1, Female = 0
  - Normalize the data

Estimator



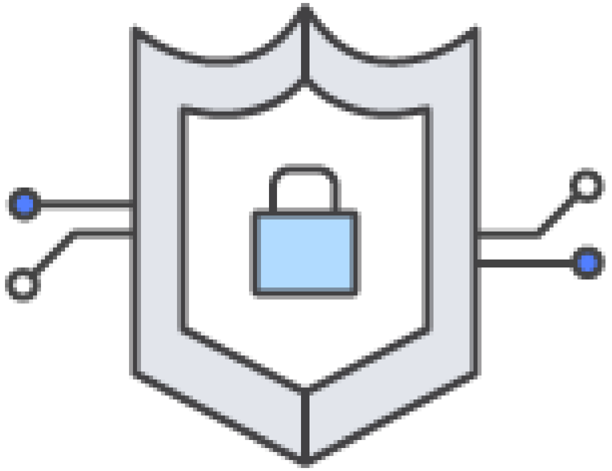
- It is a kind of transformer
- Algorithm that trains (does fit) on the data
- Resulting model is a type of transformer
- For example:
  - LogisticRegression.fit() →LogisticRegression Model
  - The StringIndexer is a type of estimator

Evaluator



- Evaluate model performance
- Assists in automating model tuning process:
  - Select best model for making predictions
  - Compare model performance

Binary Classification Project



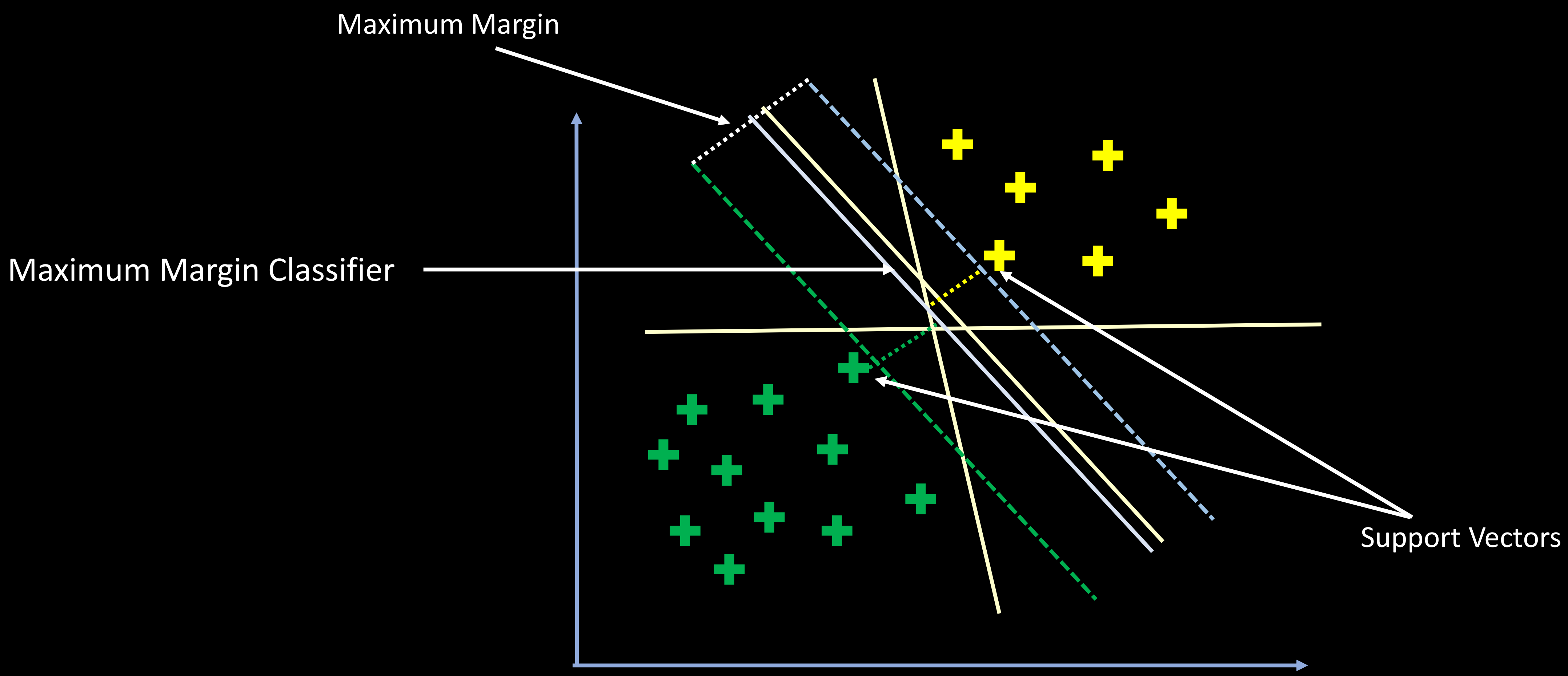
- Open file ‘SparkLab/Spark Project-Binary Classification’ using Jupyter
- Use “agent.csv” dataset containing number of features related to individuals – make prediction whether their salary is going to <= 50K or > 50K
- Using String Indexers, One Hot Encoding to process the data
- Use pipeline to process the data
- Apply Logistic Regression and make predictions

Classification: Support Vector Machine

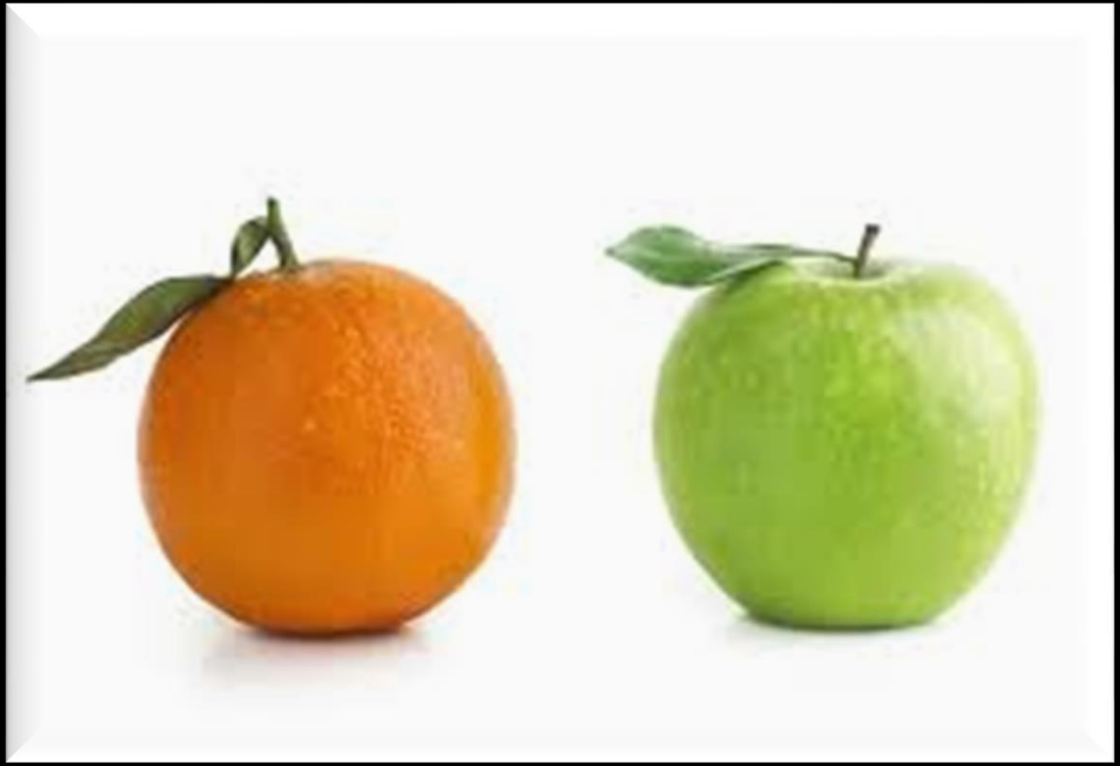
Support Vector Machine

- Support Vector Machine (SVM) is a supervised machine learning algorithm
- Supports both Classification and Regression, however it is primarily used for Classification

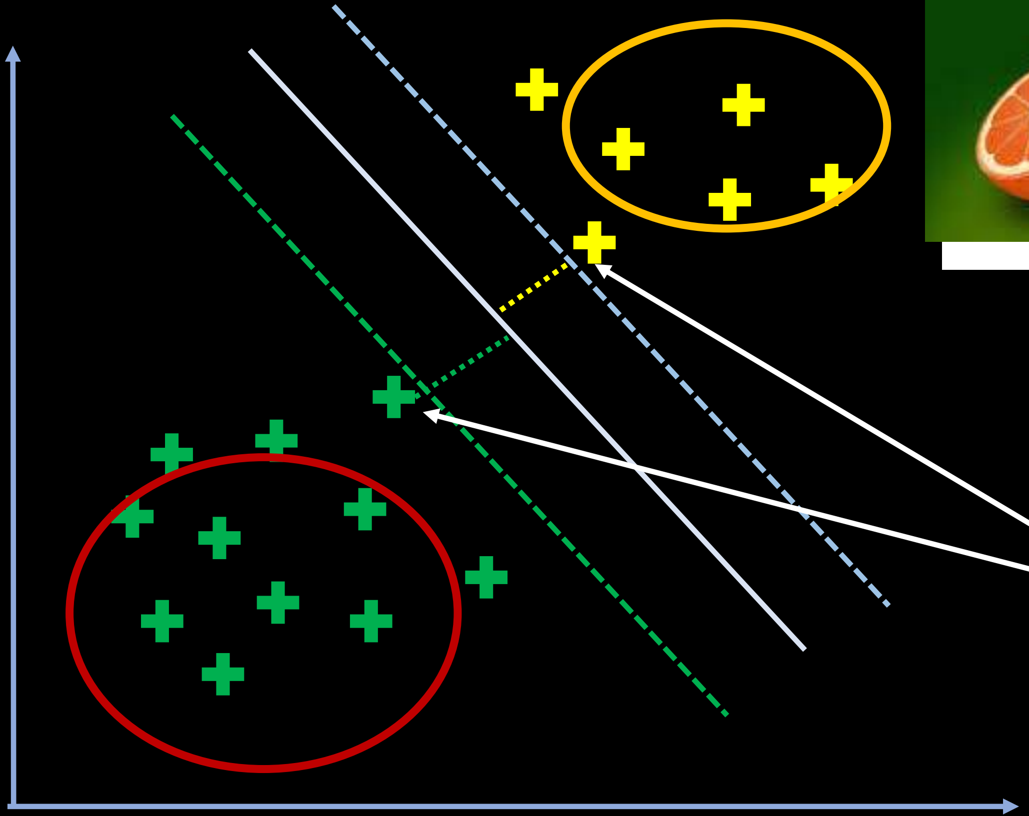
SVM – How does it work?



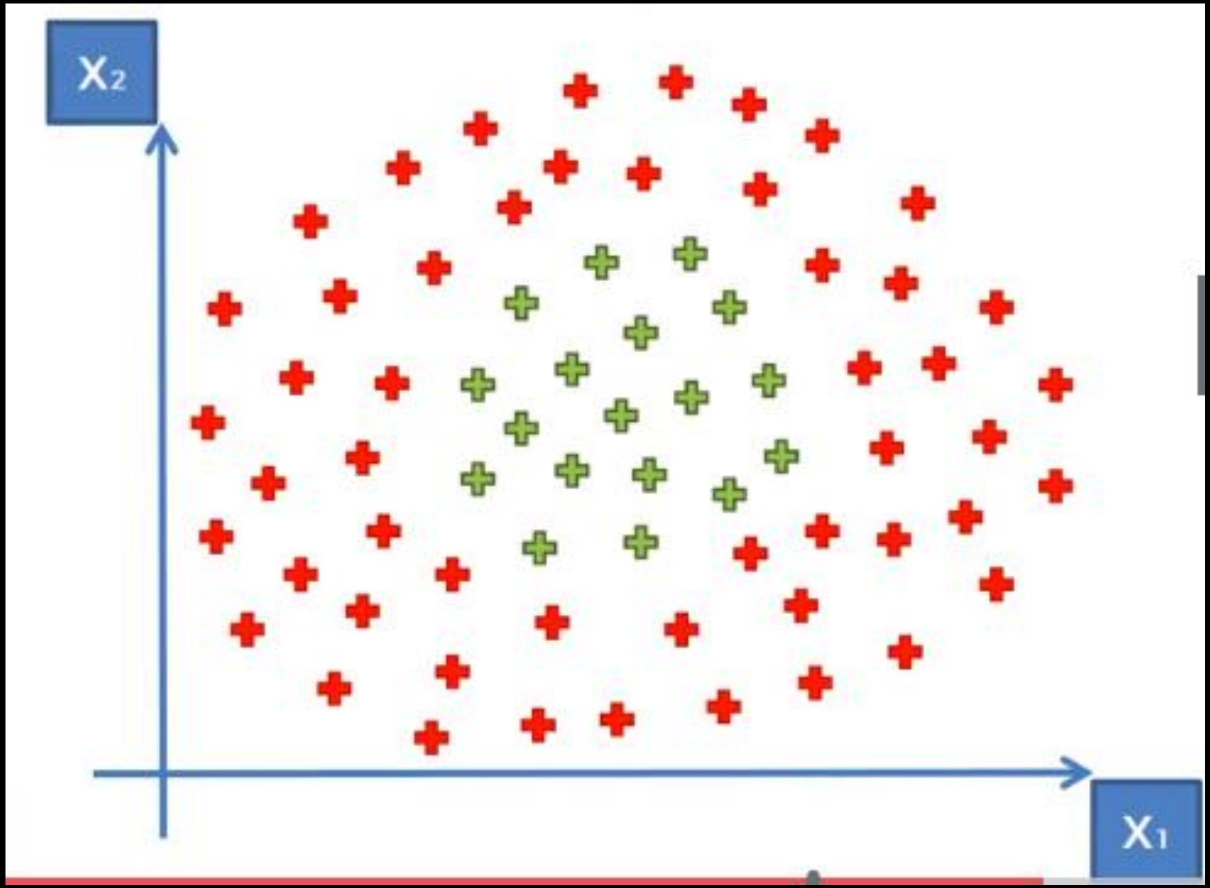
Support Vector Machine (SVM)

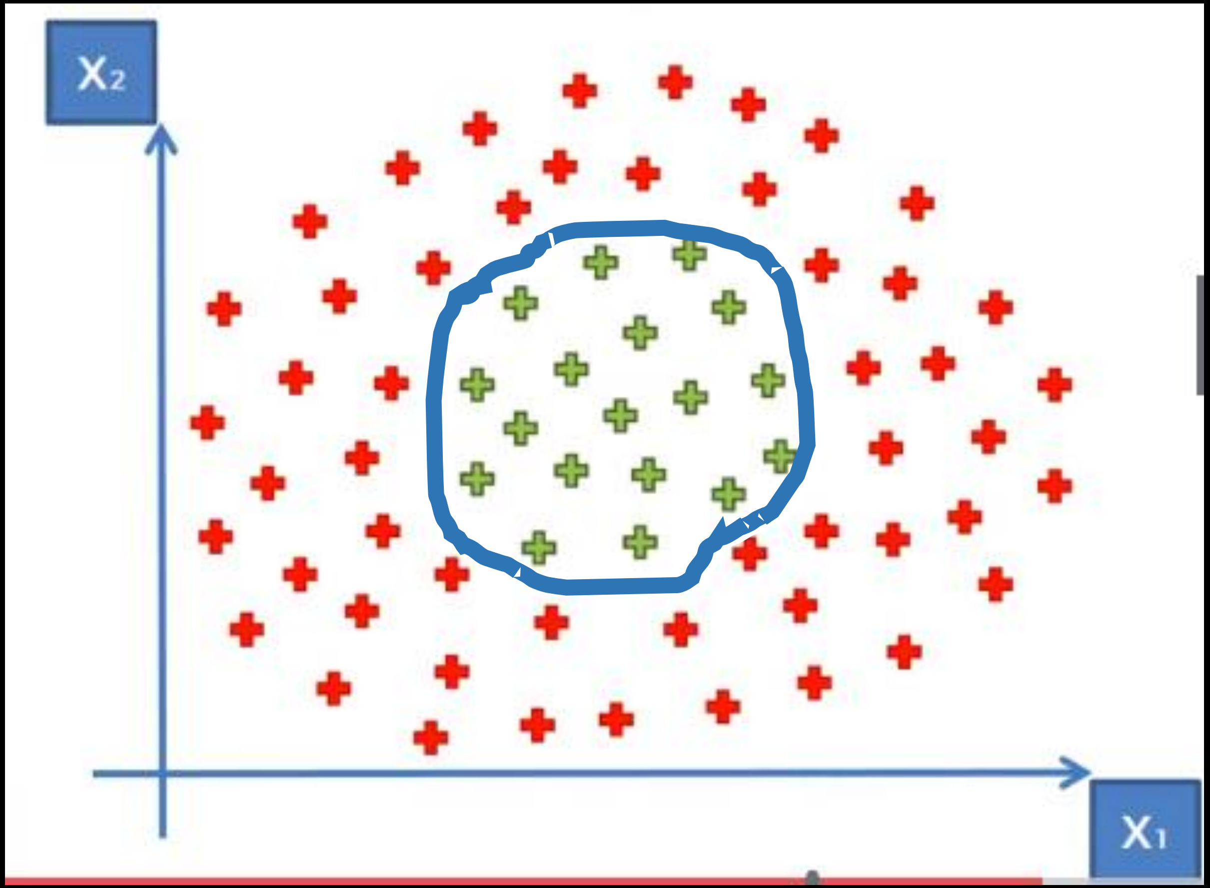


Support Vector Machines (SVM)

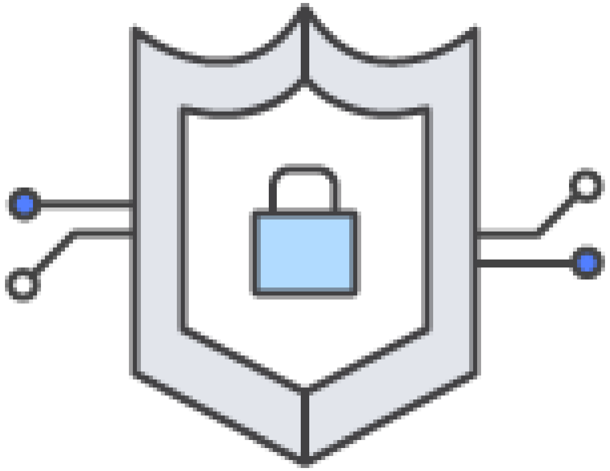


Kernel – SVM: Non-Linear Data





SVM Classification Demo



- Open file ‘SparkExamples/SVM-Classification Spark’ using Jupyter
- Use the Social Networking Ads data
- Apply support vector machine (classification) algorithm on the data
- Make predictions
- Explore training dataset metrics
- Apply binary classification evaluator

Real Life Use case : Customer Churn

Use case: Financial Customer Churn Data

- Dataset – Customer information with a financial institution
- If doing classification with SVM then it can be applied to most of the datasets where logistic regression is used
- Dataset has following features:

RowNumber: Dataset row number
CustomerId: Customer Id
Surname: Last name of the person
CreditScore: Credit Score of the person
Geography: Country of residence
Gender: Person's Gender
AGE: Age of the person
Tenure: How long has the person owned the card
Balance: Outstanding balance
NumOfProducts: Number of products owned by the person with company
HasCrCard: Person has credit card
IsActiveMember: Is the person active member of the company
EstimatedSalary: Estimated salary of the person
Exited: Did the person stay or leave

- Dataset - Churn_Modelling.csv

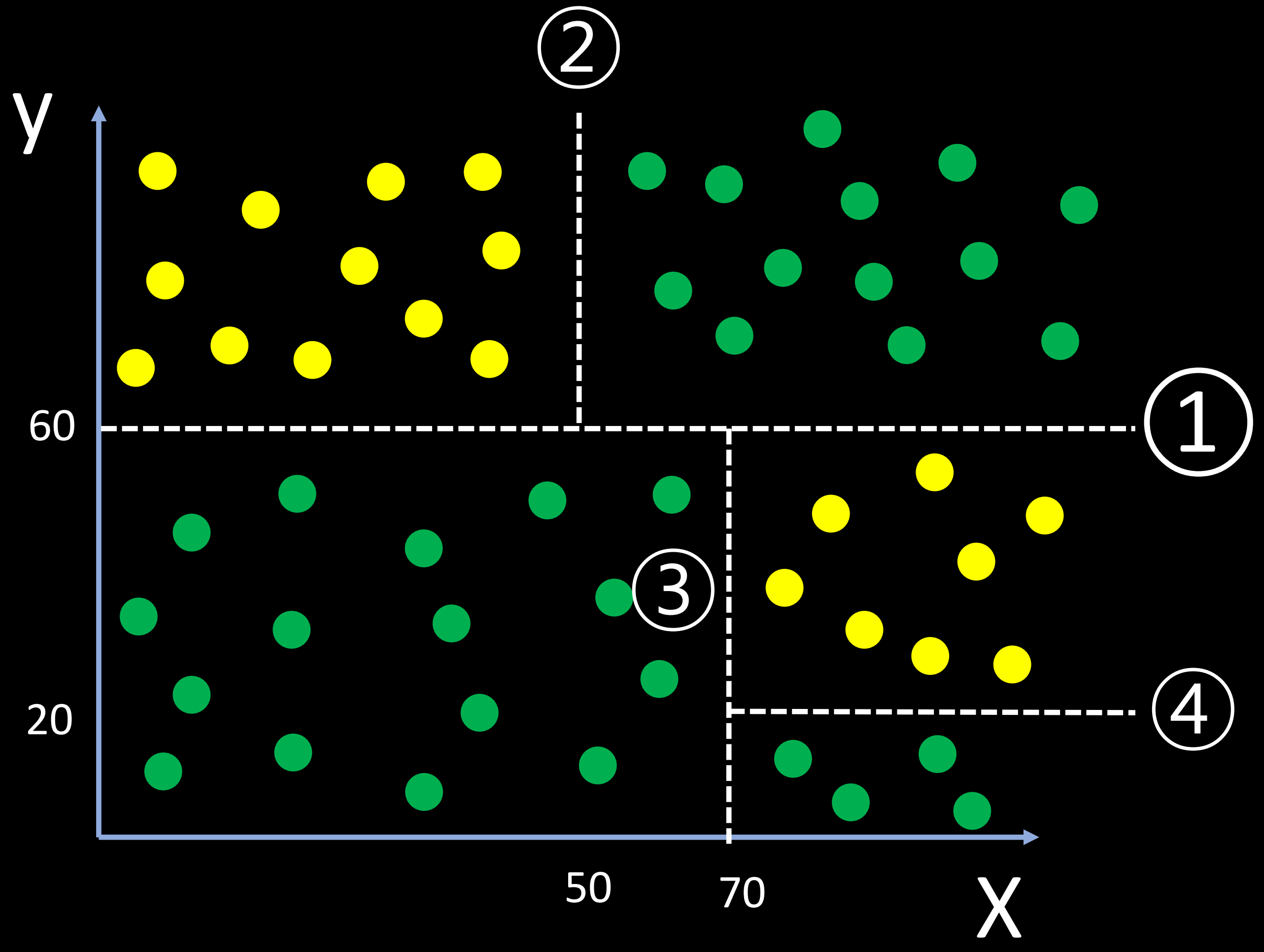
Classification: Decision Trees

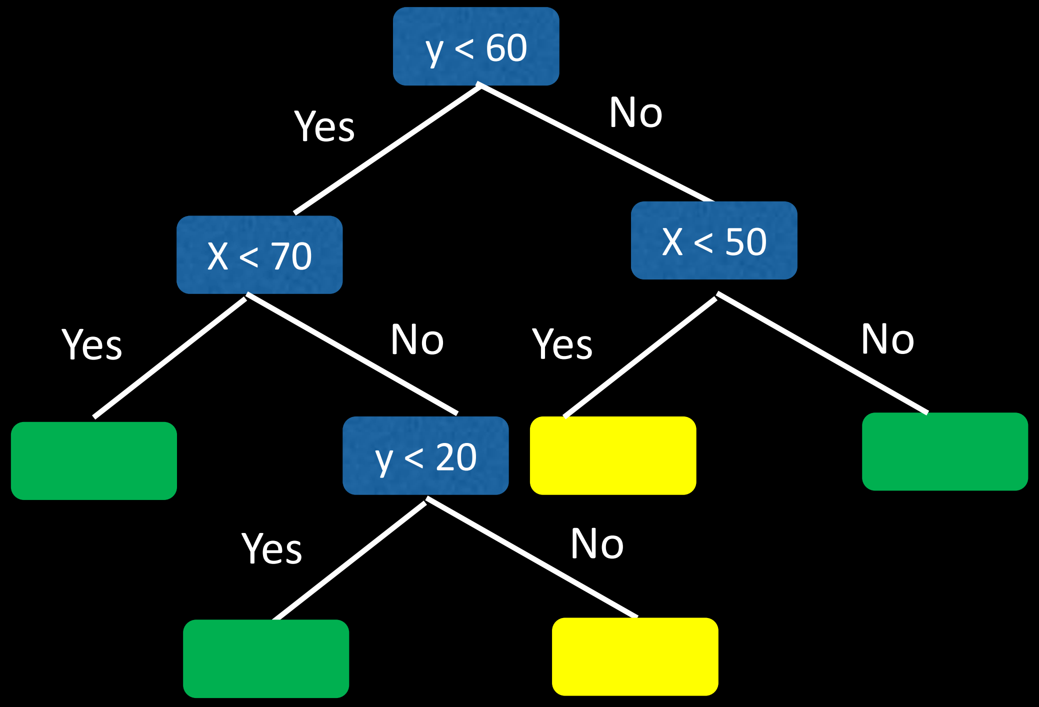
Decision Trees

Classification and Regression Tree (CART)

- CART model is a binary tree
- Classification Trees
  - Classify data into categorical variables e.g. buy not buy
- Regression Trees
  - Predict outcome that can be real numbers e.g. temperature, person's salary, etc.
- Examine Classification Trees

Decision Trees

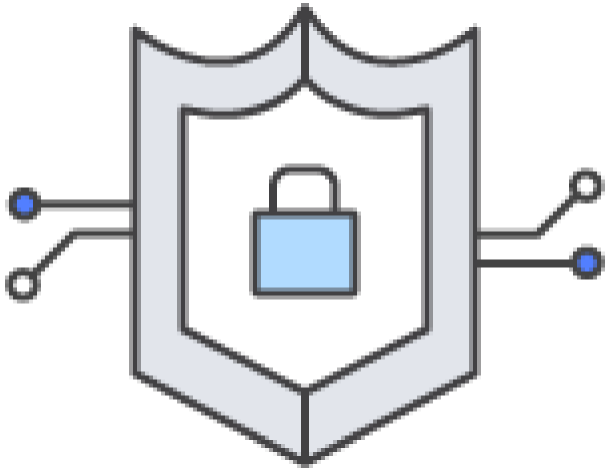




Decision Trees - Splits

- Cost function is used to determine the splits
- Regression Trees uses mean squared error (MSE) function
- Classification Trees use Gini cost function – Gini Index
- Measure of how good the split is
- A perfect separation will have value of 0, whereas 50-50 split will be the worst (value 0.5)

Decision Tree Demo



- Open file 'SparkExamples/DecisionTree Regression Spark' using Jupyter
- Using very simple and small dataset containing years of experience v/s salary
- Apply Decision Tree Regressor to predict salary based on years of experience

Decision Tree Classification - Project

Assignment

- Open notebook – "SparkLab/Spark Project-Binary Classification"
- Look for implementation of Decision Tree
- Write code for Decision Tree (Marked in Red)
- Do not worry about Bonus Implementation (ParamGrid and Cross Validator)

Classification: Random Forest

Random Forest

- Random Forest is an ensemble classifier that uses multiple decision trees
- Ensemble Models
  - Results from multiple models are combined
  - This result is better than results from individual models
  - Each individual tree predicts a best class, final result is based on taking majority votes for a class

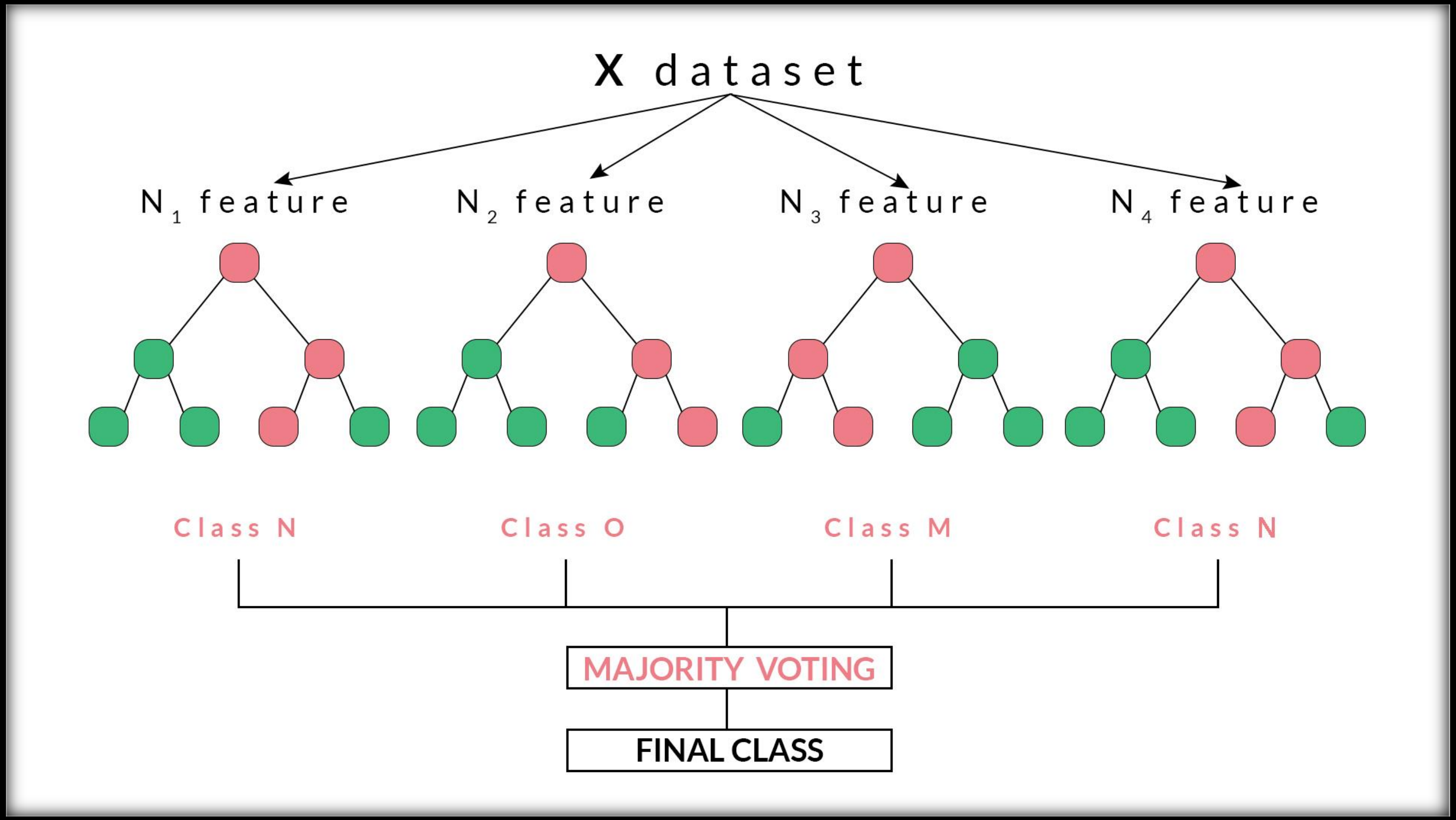
Random Forest

Pick n data points from the training set

Build a decision tree using these n points

Choose N number of trees & repeat above 2 steps

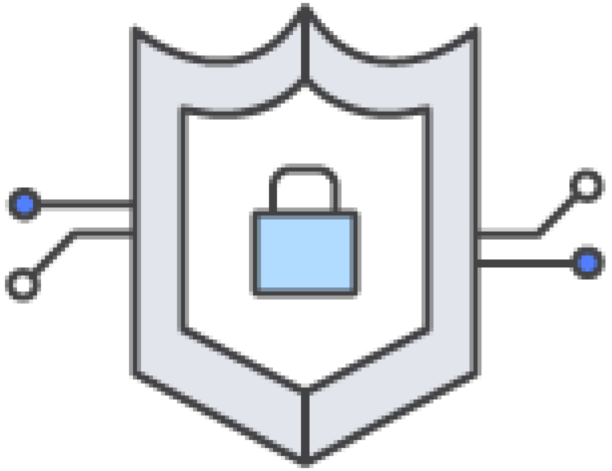
Take new data point and apply it to all N trees and get prediction. Class with majority WINS



Random Forest

- Microsoft used it in Kinect
- https://www.microsoft.com/en-us/research/wpcontent/uploads/2016/02/BodyPartRecognition.pdf

Random Forest Demo



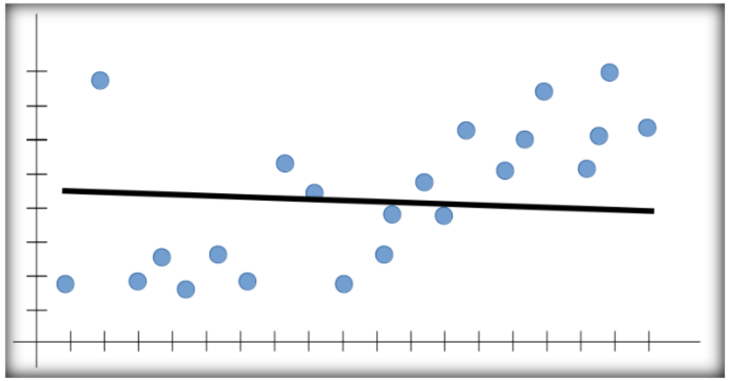
- Open file 'SparkExamples/RandomForest Spark' using Jupyter
- Using very simple and small dataset containing years of experience v/s salary
- Apply Random Forest Regressor to predict salary based on years of experience

Random Forest Classifier - Project

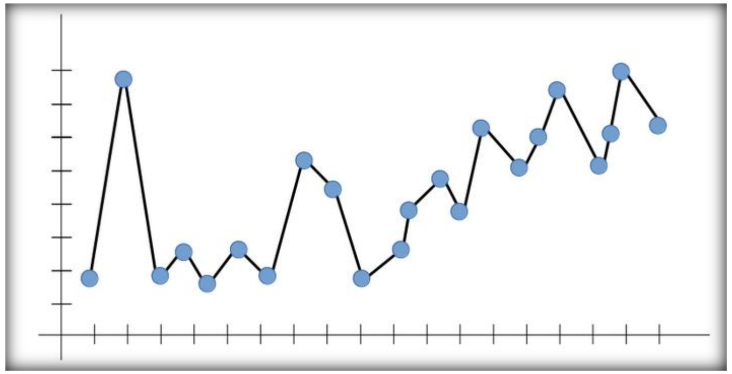
Assignment

- Open notebook – “SparkLab/Spark Project-Binary Classification”
- Look for implementation of Random Forest
- Write code for Random Forest (Marked in Red)
- Do not worry about Bonus Implementation (ParamGrid and Cross Validator)

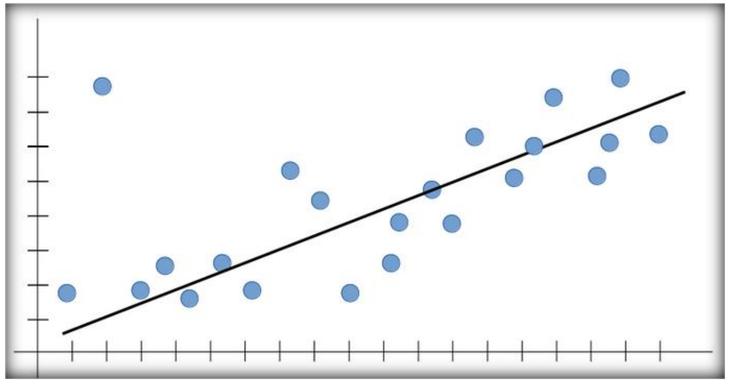
Project Model Evaluation and Tuning



Under fitting (High Bias)



Over fitting (High Variance)



Bias-Variance Trade off

Under and Over Fitting

Evaluation Model

- Evaluation is an estimate of how well our model is performing
- However, it is not a guarantee of performance
- Want to see techniques to create useful estimates of performance
- Have already seen train-test split
- *K*-fold Cross validation is another one

K-Fold Cross Validation

Training Set										
1	2	3	4	5	6	7	8	9	10	.69
1	2	3	4	5	6	7	8	9	10	.64
1	2	3	4	5	6	7	8	9	10	.73
1	2	3	4	5	6	7	8	9	10	.82
1	2	3	4	5	6	7	8	9	10	.64
1	2	3	4	5	6	7	8	9	10	.70
1	2	3	4	5	6	7	8	9	10	.68
1	2	3	4	5	6	7	8	9	10	.71
1	2	3	4	5	6	7	8	9	10	.70
1	2	3	4	5	6	7	8	9	10	.69

K-Fold Cross Validation

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	Mean
.69	.64	.73	.82	.64	.70	.68	.71	.70	.69	.70

Best Result

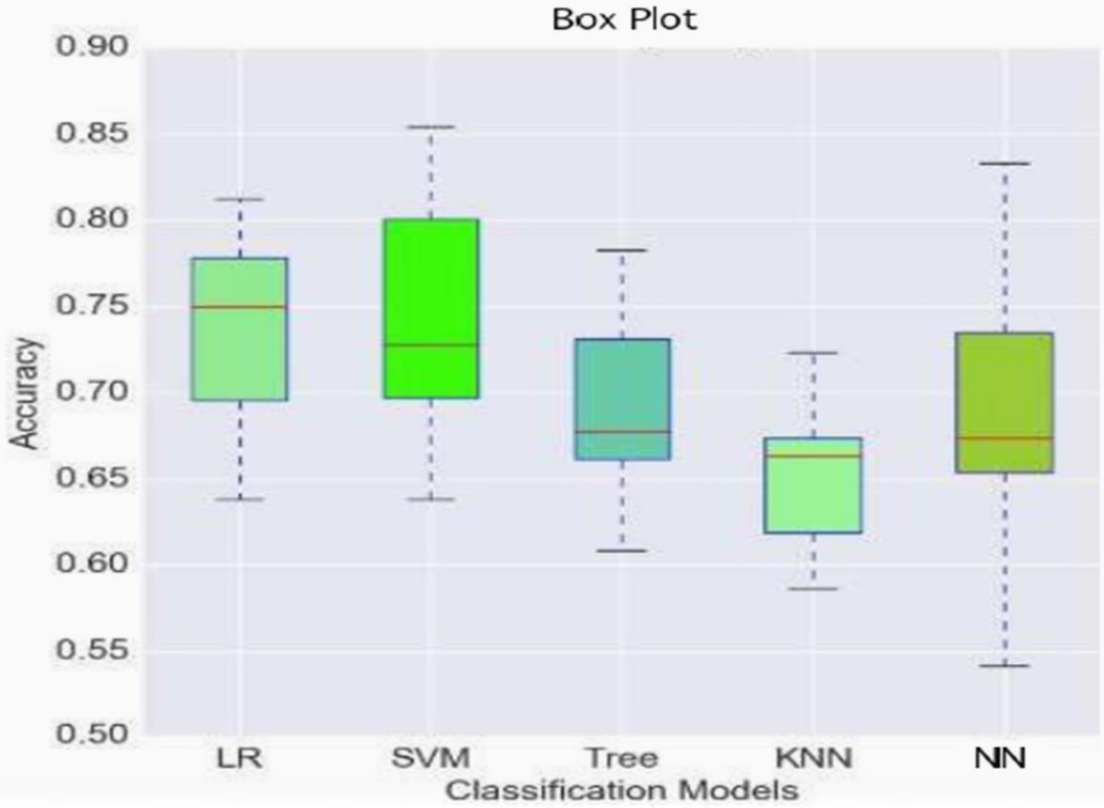
Worst Result

Improving Performance

- Select multiple algorithms on the dataset and measure performance of each
- **Parameters Tuning**
  - **Model Parameter** – these are learnt by the model and are internal
  - **Hyper Parameter** – these are like "knobs" that we use to tune the algorithm
    - Selecting number of clusters, tree depth, how many iterations, number of trees in random forest, etc.

Model Plot

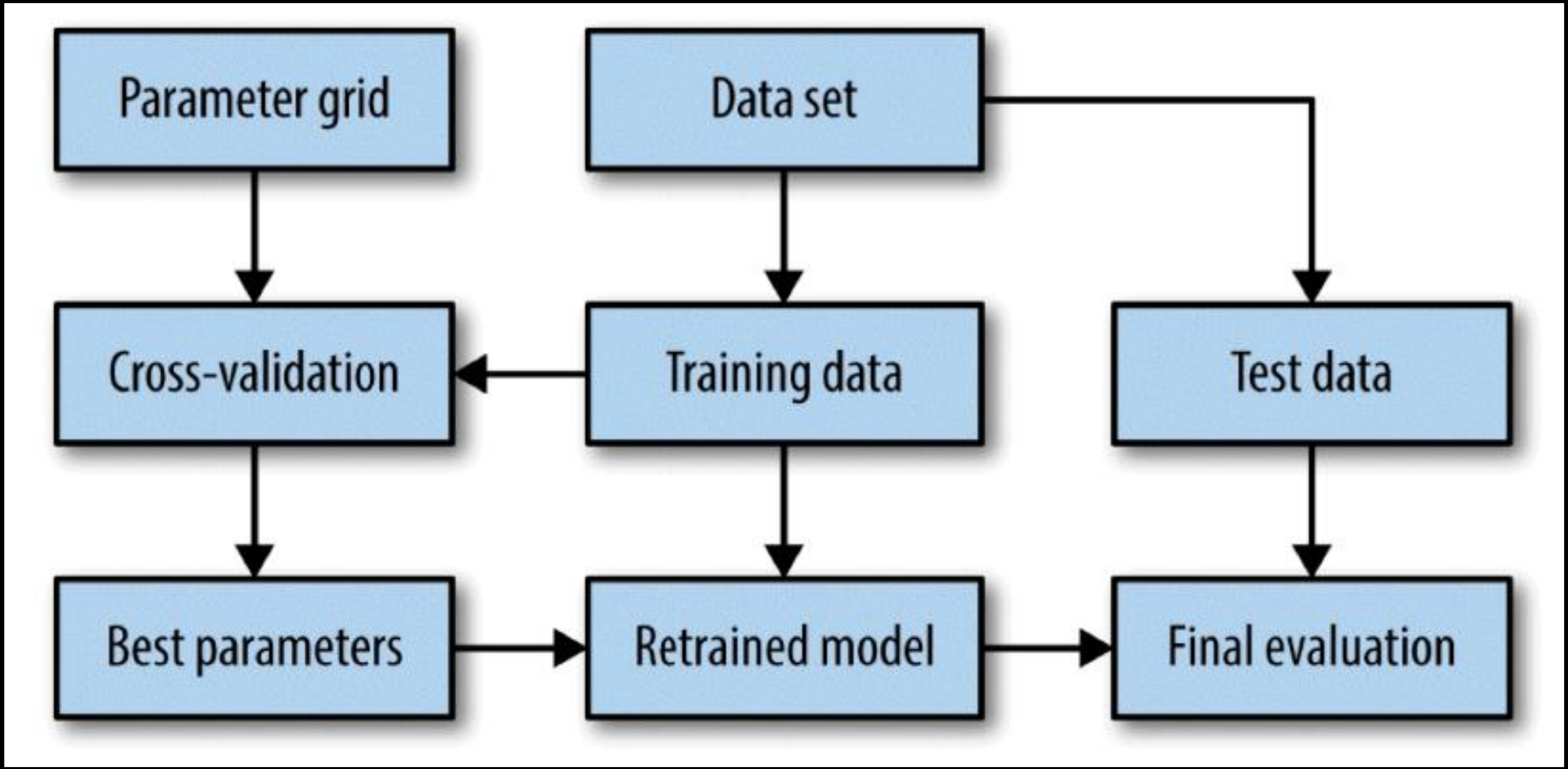
Logistic Regression	Support Vector Machine	Decision Tree	K-Nearest Neighbor	Neural Network
0.705	0.722	0.635	0.675	0.607



Grid Search – Hyper Parameter tuning

- Grid search will build and evaluate a model for each combination of algorithm parameters specified
- Example in the “Spark Project – Binary Classification”
- Implementation of “ParamGridBuilder” and “CrossValidator”

Model Evaluation – Summary Takeaway



"Introduction to Machine Learning with Python", Müller and Guido

Extra Credits

Extra Credits

- There are 3 datasets:
  1. Churn Modelling
  2. Credit Card Application
  3. Heart Disease
- Your Work
  - Pick anyone of these datasets
  - Apply the techniques that you have learnt related to Feature Cleaning, Feature selection, etc.
  - Choose 1 or more algorithms and make predictions
  - Tomorrow we will review it with the class
- Description of datasets in following slides

Use case: Financial Customer Churn Data

- Dataset – Customer information with a financial institution
- If doing classification with SVM then it can be applied to most of the datasets where logistic regression is used
- Dataset has following features:

RowNumber: Dataset row number
CustomerId: Customer Id
Surname: Last name of the person
CreditScore: Credit Score of the person
Geography: Country of residence
Gender: Person's Gender
AGE: Age of the person
Tenure: How long has the person owned the card
Balance: Outstanding balance
NumOfProducts: Number of products owned by the person with company
HasCrCard: Person has credit card
IsActiveMember: Is the person active member of the company
EstimatedSalary: Estimated salary of the person
Exited: Did the person stay or leave

- Dataset - Churn_Modelling.csv

Use Case: Credit Card Application

- Based on 15 features (not named), predict whether a new customer’s credit card application should be approved
- Predicting a “class” – 1 or 0
- Dataset - Credit_Card_Applications.csv

Use Case: Heart Disease

- Based on 13 features predict if a patient will have heart disease
- Features include age, gender, chest pain, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise induced angina, etc.
- Predicting a “target” – 1 or 0
- Dataset - heart.csv

Unsupervised Learning

Clustering

- We have unlabeled data
- No predefined output classes. We can only group the data into clusters based on the characteristics of input data
- Output is dynamic based on input values, a new set of input values could change the output
- This type of Machine Learning is known as:
- ***UNSUPERVISED LEARNING***

K-Means Clustering

1. Initialization

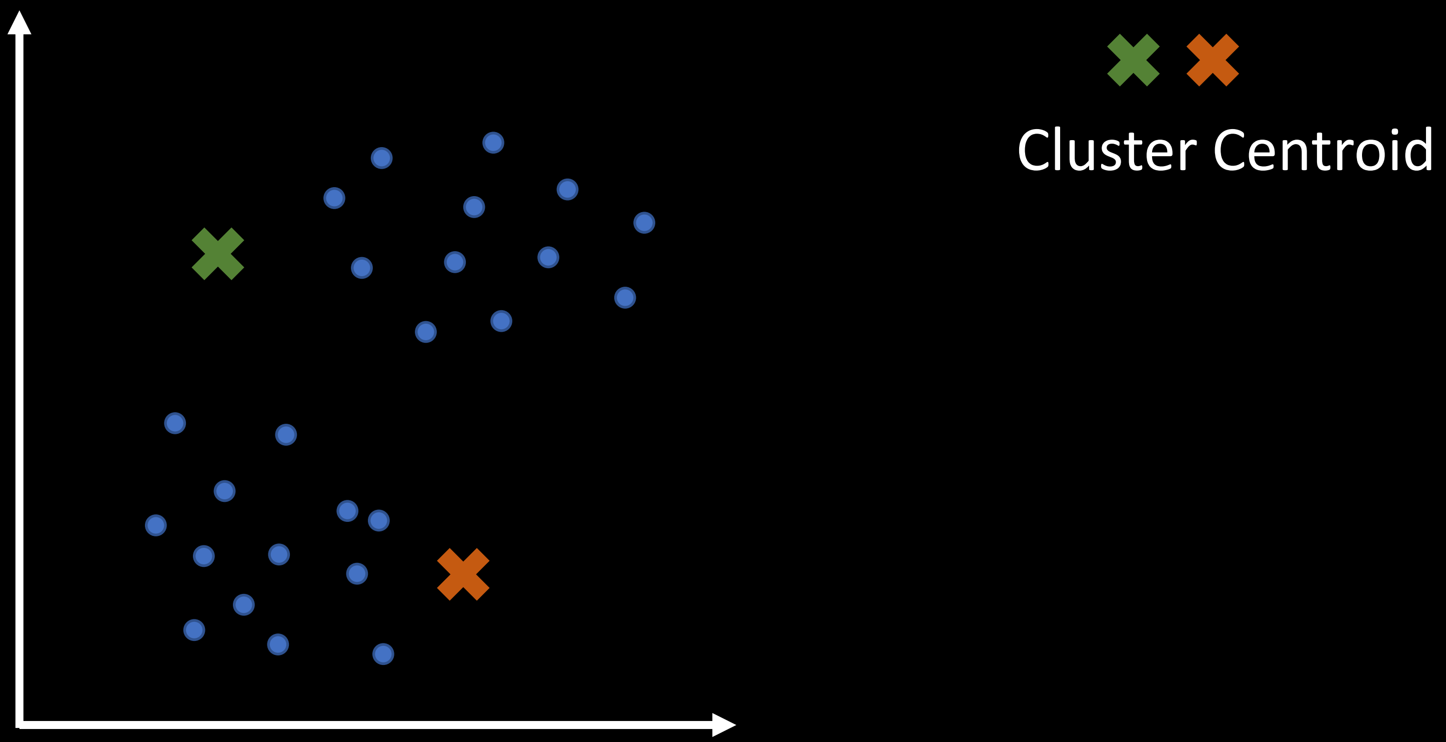
2. Cluster Assignment

3. Adjust Centroid

4. Optimize

5. Converge

- Select the number of clusters
- Randomly select k points called “Cluster Centroid”
- Example k = 2



K-Means Clustering

1. Initialization

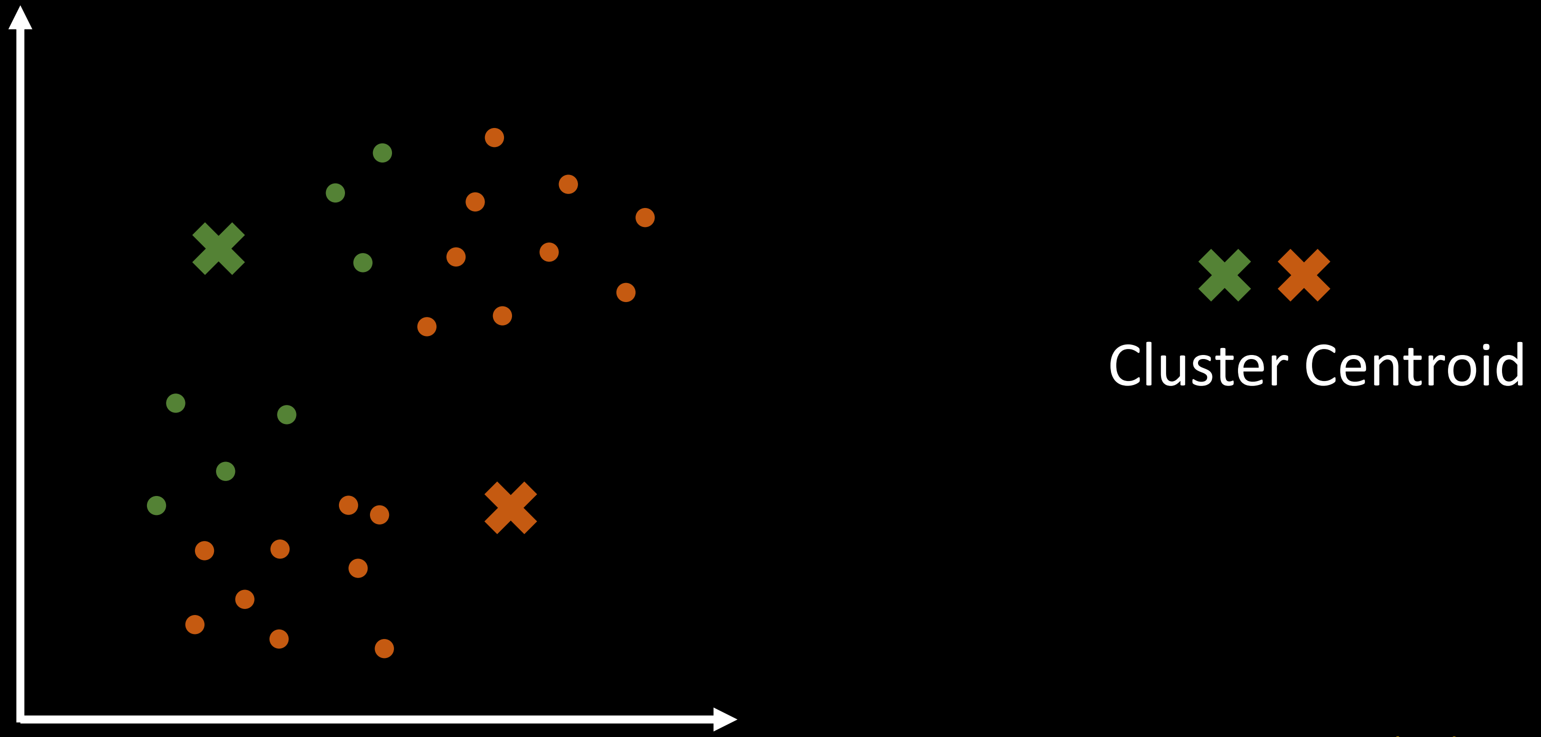
2. Cluster Assignment

3. Adjust Centroid

4. Optimize

5. Converge

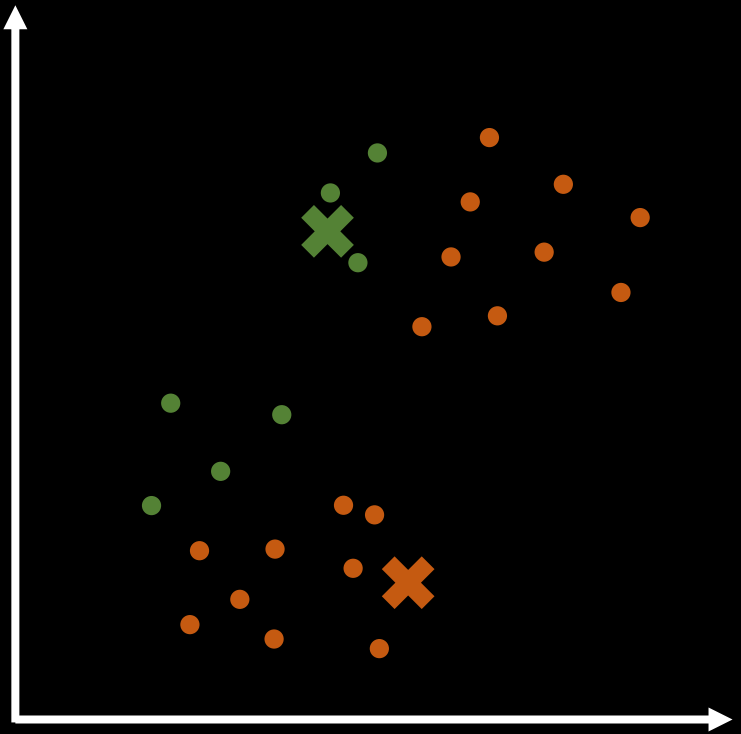
- Calculate the Euclidean distance between the data points and the centroids
- Based on minimum distance, divide data points into two groups



K-Means Clustering

1. Initialization
2. Cluster Assignment
3. Adjust Centroid
4. Optimize
5. Converge

- Compute mean for green dots, reposition the green cluster centroid to this mean
- Compute mean for orange dots, reposition the orange cluster centroid to this mean

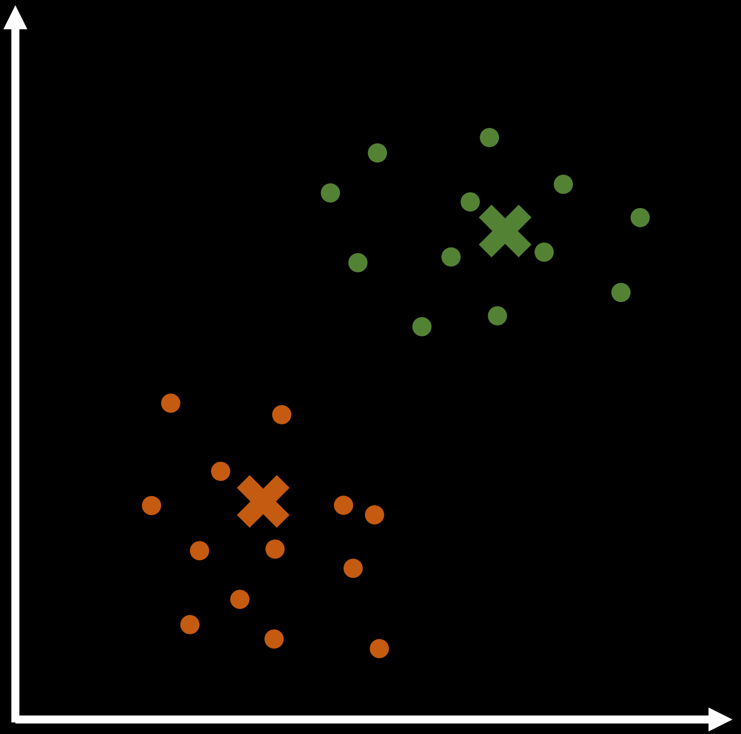


Cluster Centroid

K-Means Clustering

- Repeat this process until the cluster centroids stop changing their positions

1. Initialization
2. Cluster Assignment
3. Adjust Centroid
4. Optimize
5. Converge



K-Means Clustering

1. Initialization

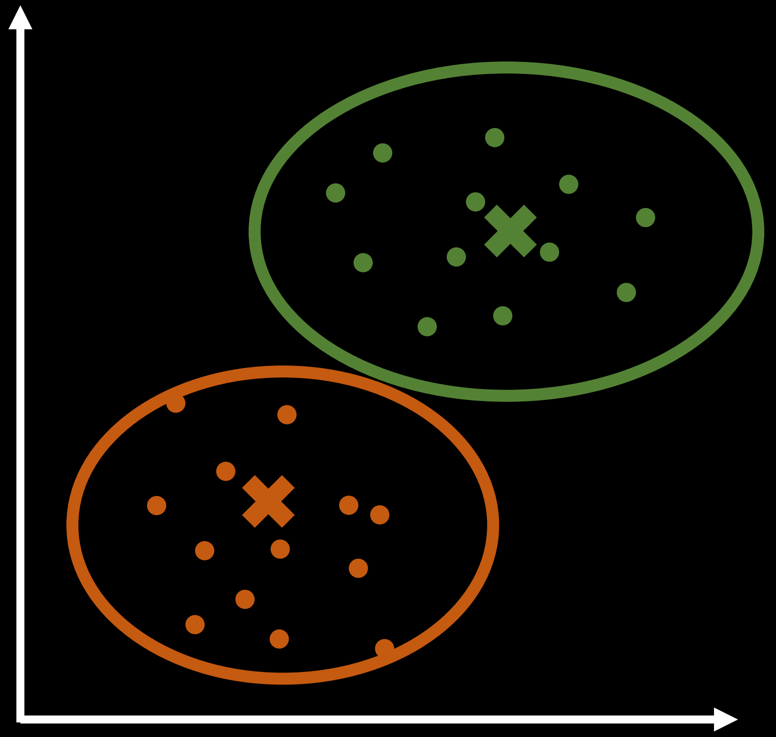
2. Cluster Assignment

3. Adjust Centroid

4. Optimize

5. Converge

- Finally the model converges
- We have 2 clusters

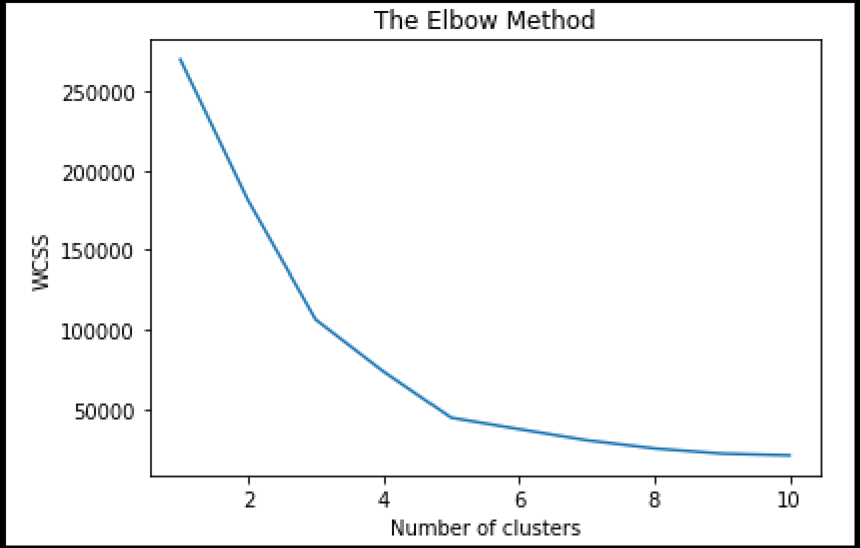


Cluster Centroid

Choosing the number of clusters

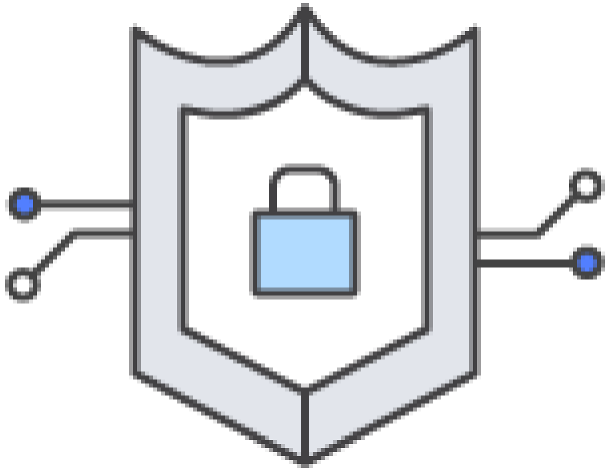
Elbow Method:

- First calculate the sum of squared error (SSE) for values of k (e.g. 2, 4, 6, 8, etc.). It is the sum of squared distance between each point in the cluster and its centroid
- As k increases SSE gets smaller, because as k increases the distortion is smaller
- Select the number of clusters when the SSE stops dropping abruptly



Number of clusters is 5

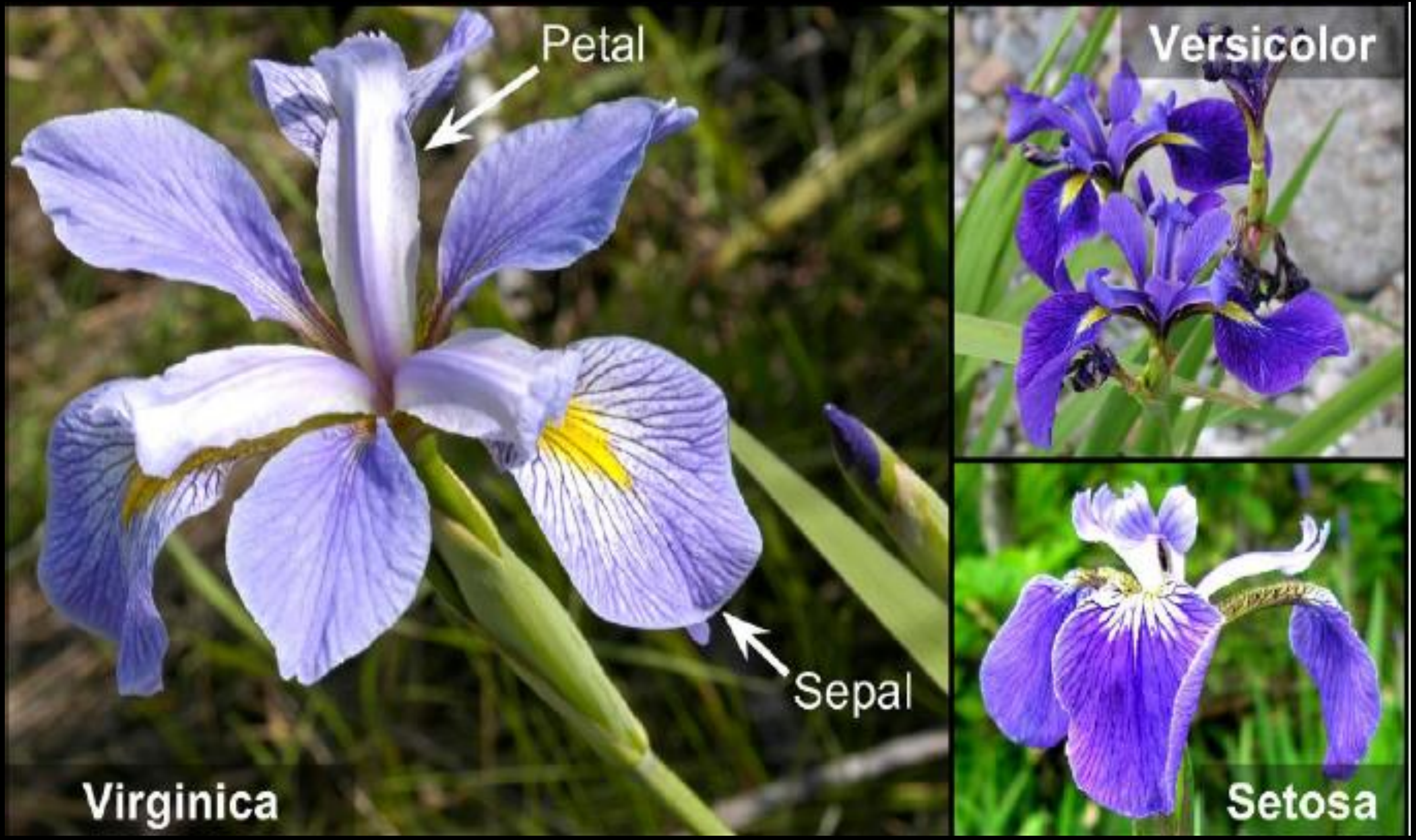
K-Means Clustering Demo

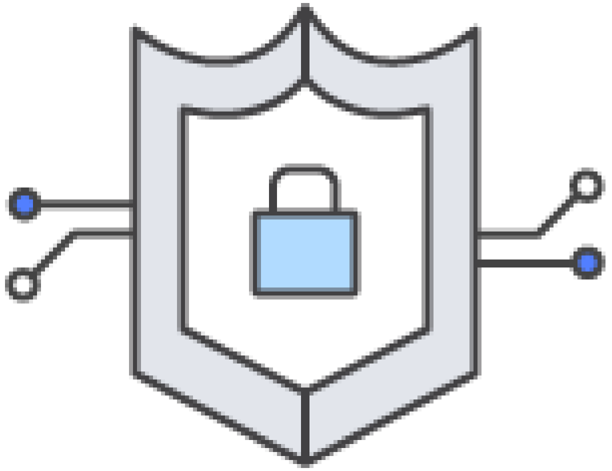


- Open file ‘SparkExamples/K-Means-Clustering Iris’ using Jupyter
- Find WCSS - Within Cluster Sum of Squares
- Use Elbow method to predict number of clusters (here we already know that there are 3 types of flowers
- Use K-Means Clustering to group the flowers – make predictions

Iris Dataset

- Sepal and Petal widths to identify types of iris flowers
- Attribute Information:
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm
  - Class/type:
    - Iris Setosa
    - Iris Versicolor
    - Iris Virginica
- Number of Samples: 150





K-Means Clustering Demo

- Open file 'SparkExamples/K-Means-Clustering Spark' using Jupyter
- Find WCSS - Within Cluster Sum of Squares
- Cluster all customers based on age, gender, annual income and spending score
- Create clustering of these customers using K-Means