ASCEND towards a sustainable future

KIRON e-Learning for Future

Python Fundamentals for Data Science

Session-6

ASCEND towards a sustainable future

KIRON e-Learning for Future

Agenda

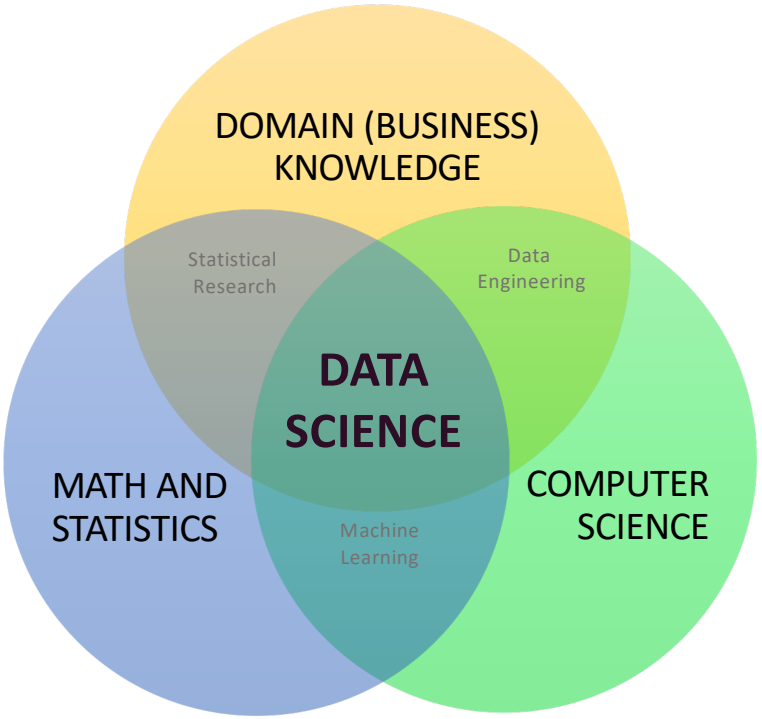
Lecture -6: Python Fundamentals for Data Science

- ❑ Understand the data science life cycle, essential tools, and industry applications
- ❑ Learn about the popular data science and ML frameworks
- ❑ Start building a basic regression model

Data Science Skills

Data Science is a field of study that combines the following skills to extract meaningful insights from data :

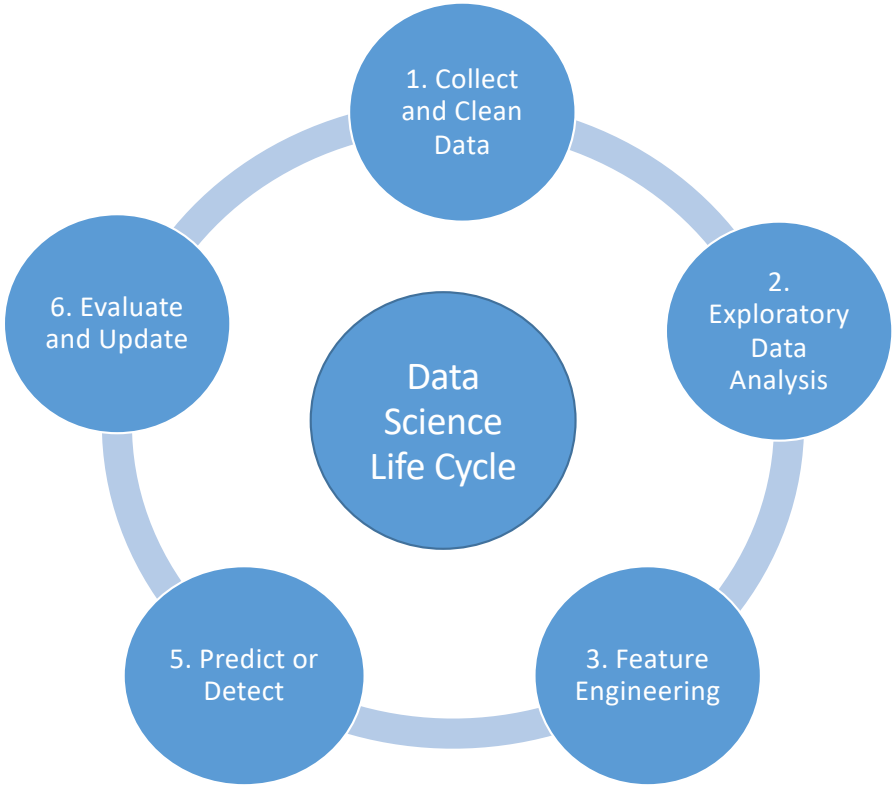
1. Business understanding
2. Programming skills, and
3. Math and statistics skills



Data Science Life Cycle

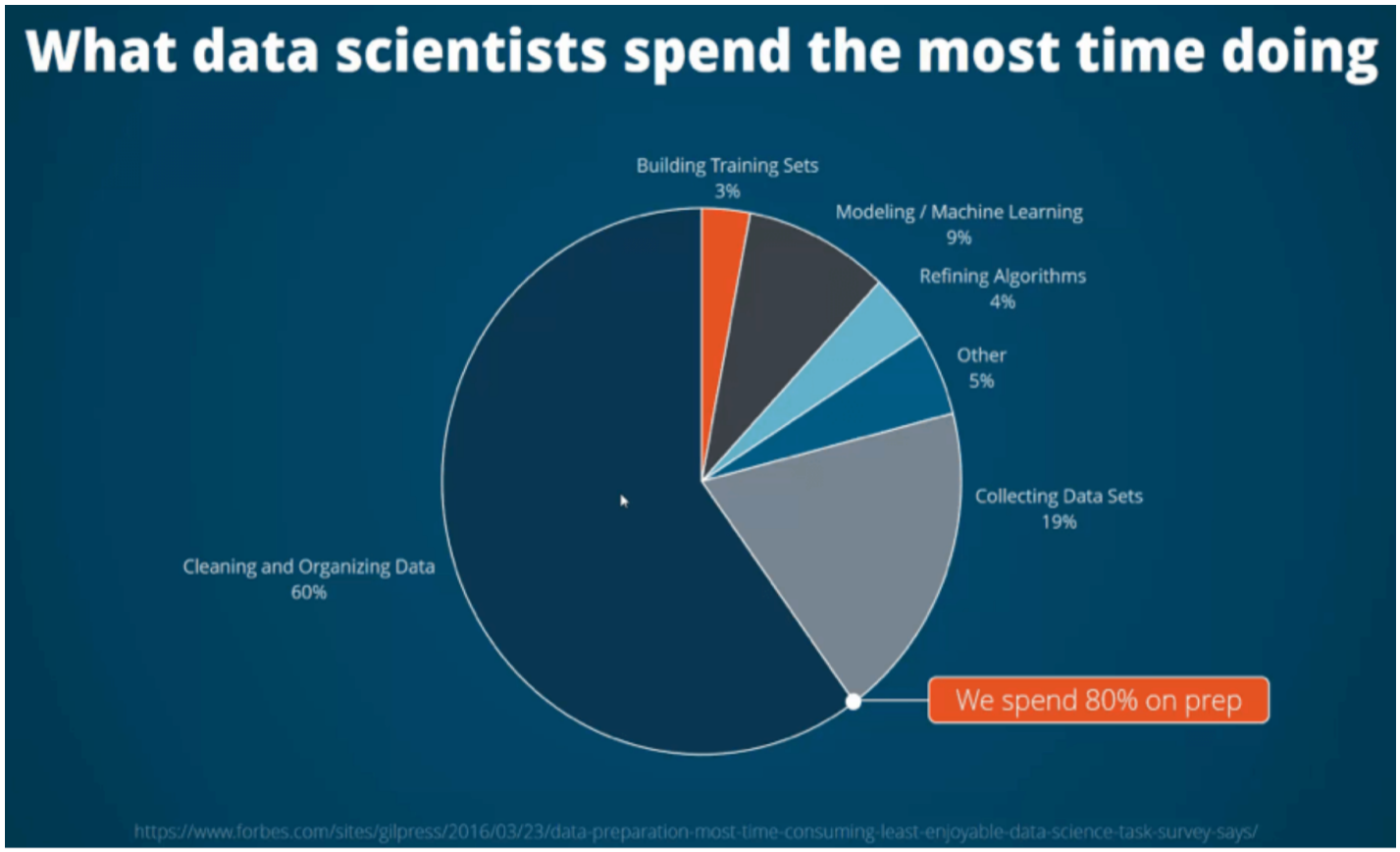
Data Science Life Cycle

- This is an iterative process and you can iterate between any of these steps to improve your work.
- For example, after evaluating the model result in step 6, you may not get a satisfactory result and you may have to go to step 3 to create a new feature
- We use data visualization in most of these steps to understand input and output data



Data Science Life Cycle

- As a Data Scientist most of your time will be spent on data preparation and pre-processing.
- For modeling and machine learning, you will spent small percentage of your time

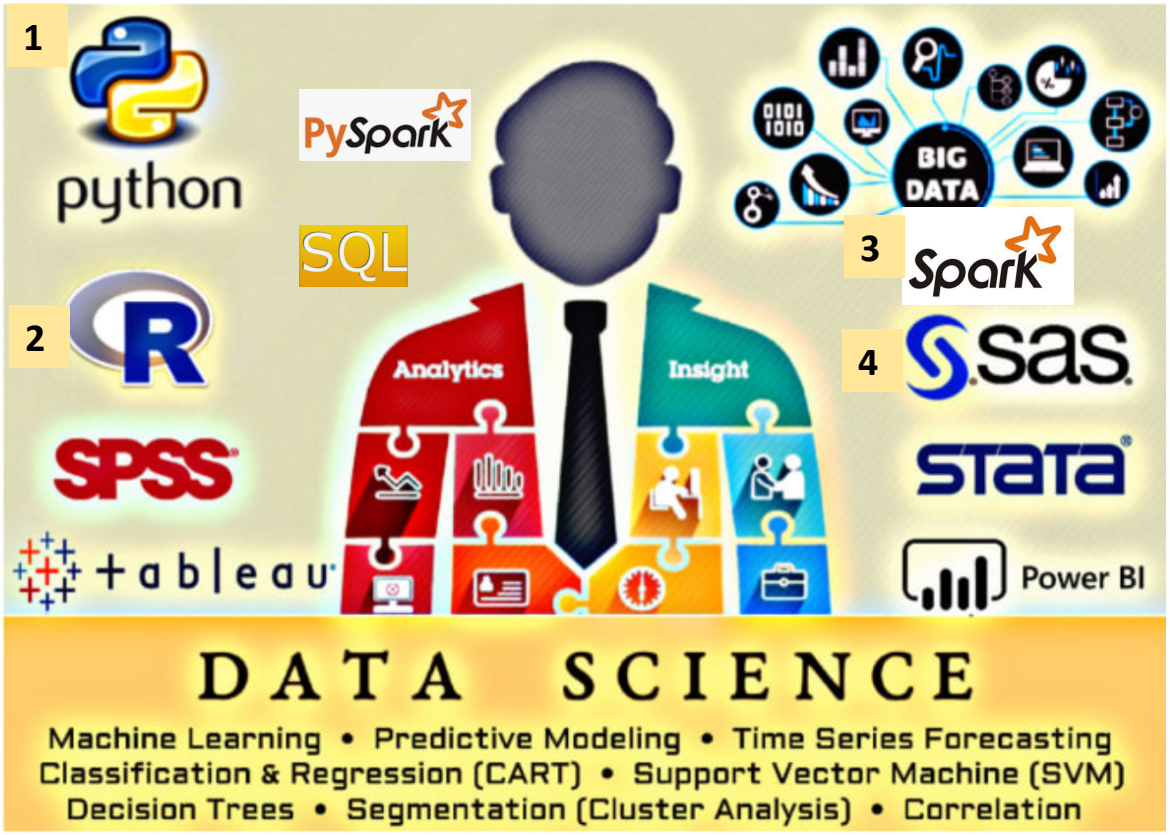


Popular Tools And Frameworks

Data Science Tools

Top 7 Data Science Tools

1. Python
2. R
3. Apache Spark
4. SAS
5. Tableau
6. Power BI
7. SQL



Popular Machine Learning Frameworks

Top 7 Data Science Frameworks

1. Scikit-Learn
2. TensorFlow
3. Pytorch
4. Keras

1. Spark MLLib
2. Theano
3. MXNet

- Scikit-Learn and Pytorch are Python native and used along with Numpy and Pandas
- Spark is used for big data processing
- We will lean and use Scikit-Learn during rest of the course

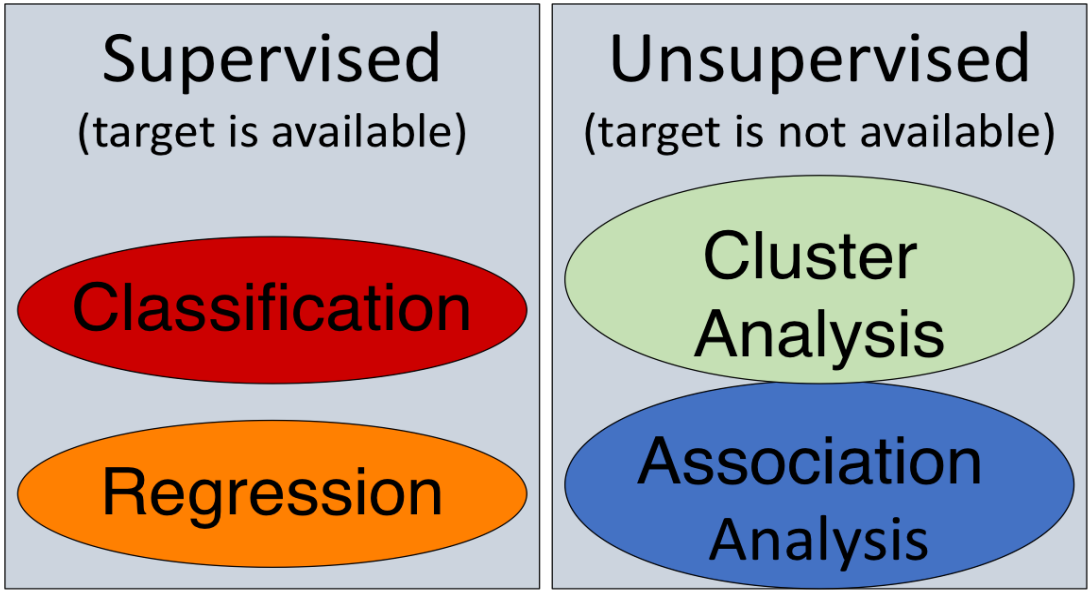
Scikit-Learn

Machine Learning With Scikit-Learn

- A Library of simple and efficient tools for data mining, machine learning and data analysis
- Built on NumPy, SciPy, and matplotlib. Accessible to everybody, and reusable in various contexts
- Open source and commercially usable under BSD license
- Extensive set of tools for full pipeline in Machine Learning

Machine Learning Categories

- Supervised: Training data is labeled
- Unsupervised: There is no label to training data



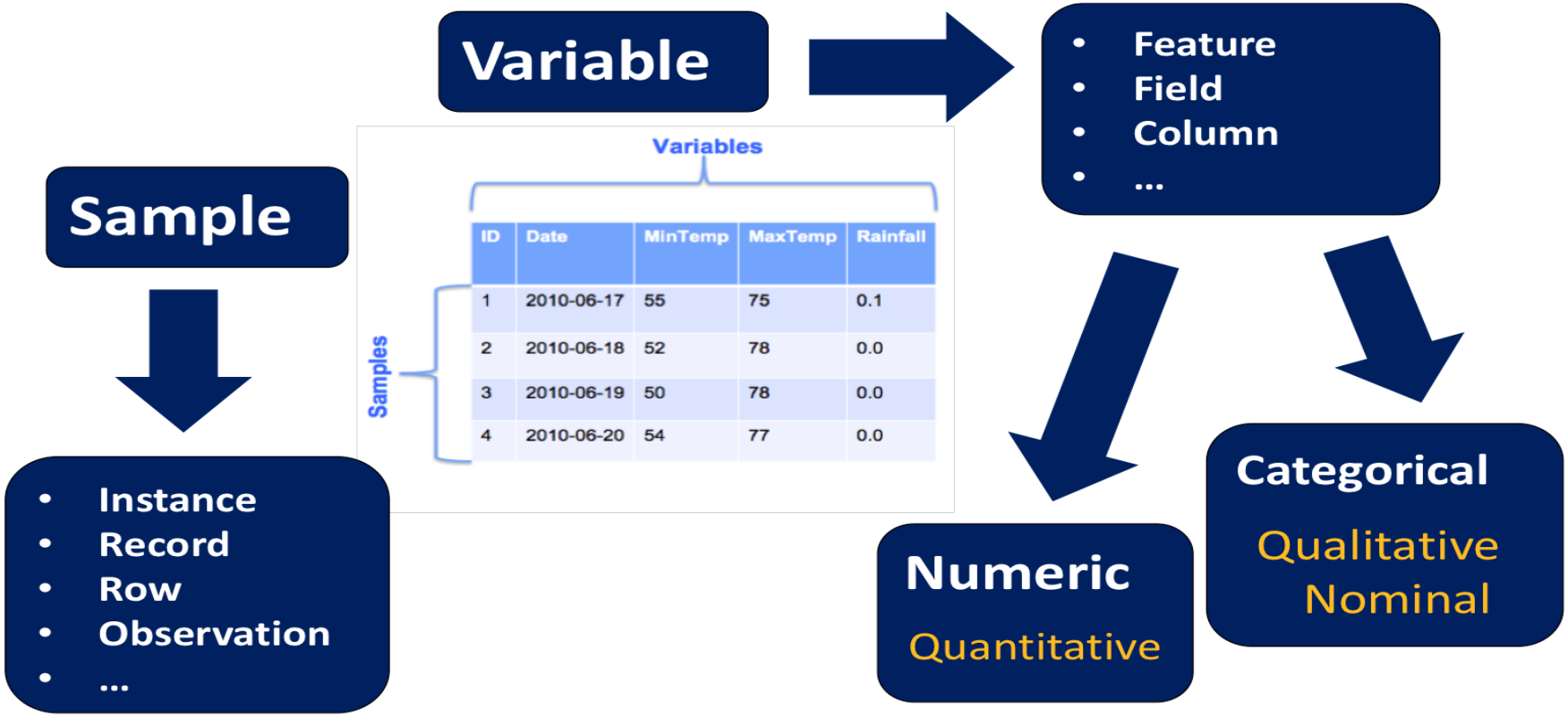
Quiz - 1

We need labeled data for a regression model as this is a supervised model.

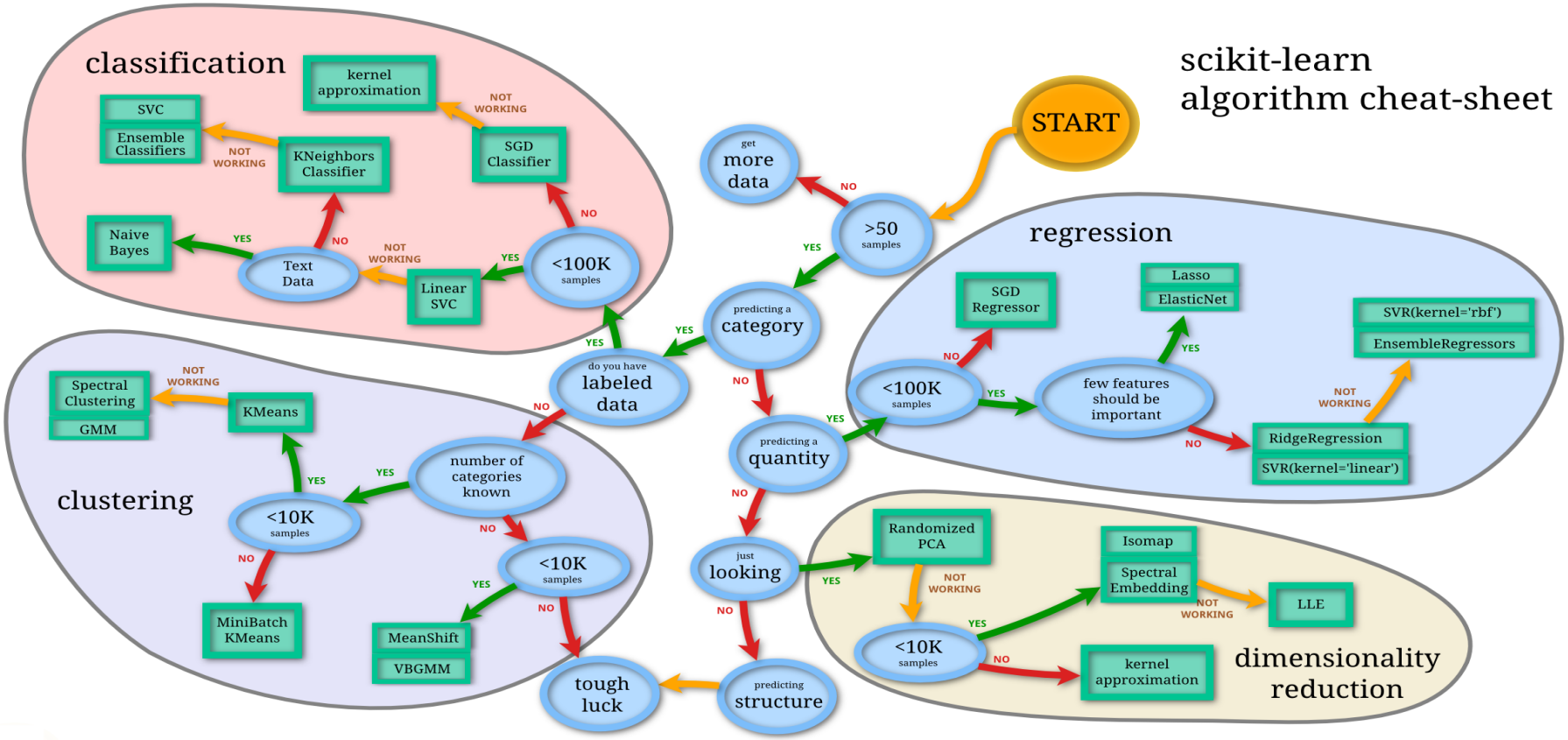
a) True

b) False

Machine Learning Terminologies



Machine Learning with Scikit-learn

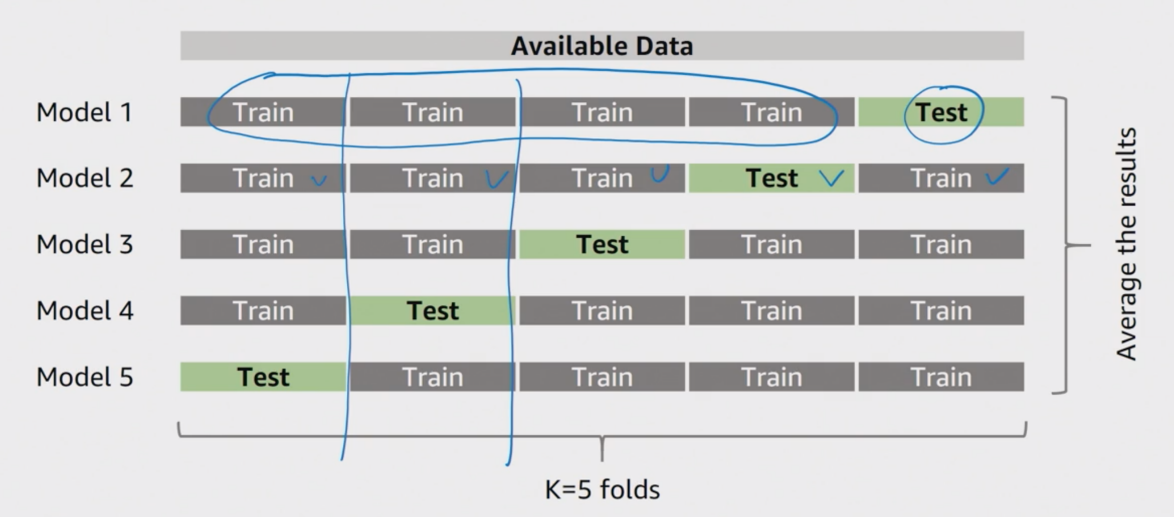


Splitting Data into Training and Test

- Before building a machine learning model we always split the data into two subsets: 1) Train and 2) Test. Sometimes we even split into three subsets – 1)Training, 2)Test and 3)Validation.
- The training subset is used for building your model. It should always be tested with some data that are not used for training.
- The testing subset is used for test the model on unknown data that is not used for training. In that way we can verify that the model will perform on the unknown data.
- Typically we use 50% to 70% data from training and 30% to 50% for testing, i.e. (70% training, 30% testing), or (60% training, 40% testing) etc.

Model Selection - Cross-validation

- Randomly splits the whole dataset k times in training and test data
- Subsets of the training set with varying sizes to train the model
- A score for each training subset and the test set will be computed
- Scores averaged over all k runs for each training subset



Choosing K

- Large - May result more variance; Compute intensive
- Small - May result more bias
- 5 to 10 is typical

Quiz -2

Repeated k-fold cross-validation provides a way to reduce the error in the estimate of mean model performance.

a) True

b) False

Model Evaluation Metrics - Regression

- The sklearn.metrics module implements several loss, score, and utility functions to measure regression performance.
- Mean_squared_error, and r2_score are most commonly used for regression models
  - Mean_squared_error

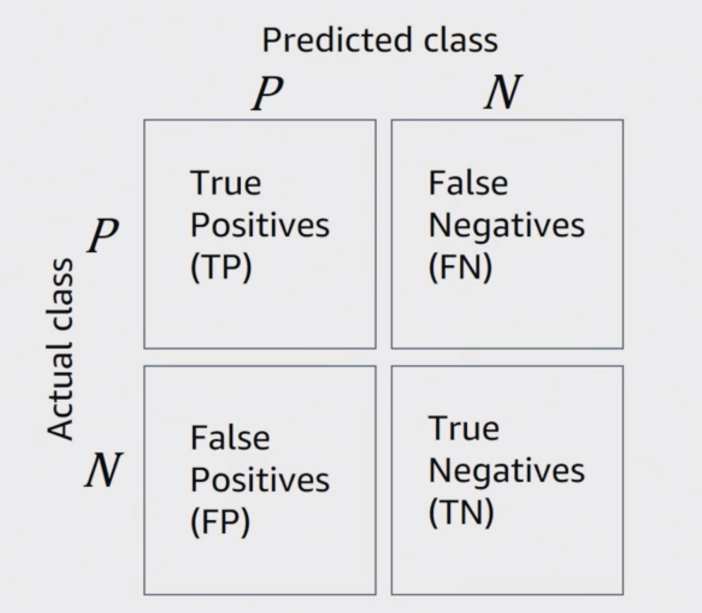
    If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over n_{samples} is defined as

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

  - r2_score
    - R-squared (R^2) is a statistical measure that represents the proportion of the variance for a dependent variable that can be explained by an independent variable or variables in a regression model.
    - Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse).

Model Evaluation Metrics - Classification

A confusion matrix is computed to evaluate the accuracy of a classification model.



True Label \ Predict Label	1 (M)	0 (B)
1 (M)	146	24
0 (B)	11	274

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

Quiz -3

Which metric do we use for a regression model?

a) Accuracy

b) Mean_squared_error

c) Both of them

d) None of them

Building A Regression Model

Scikit-Learn – Few Useful Libraries

- Datasets
- Metrics
- Model_selection
- Preprocessing
- Pipeline

Scikit-Learn – Datasets

Scikit-learn comes with a few small standard toy datasets - no need to download

They can be loaded using the following functions:

load_boston(*[, return_X_y])	Load and return the boston house-prices dataset (regression).
load_iris(*[, return_X_y, as_frame])	Load and return the iris dataset (classification).
load_diabetes(*[, return_X_y, as_frame])	Load and return the diabetes dataset (regression).
load_digits(*[, n_class, return_X_y, as_frame])	Load and return the digits dataset (classification).

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
dataset = pd.DataFrame(boston_dataset.data, columns = boston_dataset.feature_names)
```

Scikit-Learn – Datasets

- The Boston dataset contains information collected by the U.S Census Service
- It has been used extensively throughout the literature to benchmark algorithms. The dataset is small in size with only 506 cases.
- The data was originally published by Harrison, D. and Rubinfeld, D.L. `*Hedonic prices and the demand for clean air*', J. Environ. Economics & Management, vol.5, 81-102, 1978.
- We will create a regression model to predict the median value of a home

Scikit-Learn – Preprocessing

- Use the **sklearn.preprocessing** package and its Python functions to change raw input data into a representation that is more suitable for the model.
- One of them is the **Standardization** of datasets. It is a common requirement for many machine learning estimators implemented in scikit-learn.
- In general, models (learning algorithms) benefit from standardization of the data set, specially when some outliers are present in the dataset
- Model does not perform well if the individual features/inputs do not look like standard normally distributed data: Gaussian with **zero mean** and **unit variance**.

Scikit-Learn – Preprocessing

```
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> scaler
StandardScaler()

>>> scaler.mean_
array([1. ..., 0. ..., 0.33...])

>>> scaler.scale_
array([0.81..., 0.81..., 1.24...])

>>> X_scaled = scaler.transform(X_train)
>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

```
>>> X_scaled.mean(axis=0)
array([0., 0., 0.])

>>> X_scaled.std(axis=0)
array([1., 1., 1.])
```

Quiz -4

Standard normal distribution does not need to have a **mean** of zero and **standard** deviation of 1.

a) True

b) False

Scikit-Learn – Train Test Split

Train-test function splits arrays or matrices into random train and test subsets

```
# 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 = 0.3, random_state = 25)
```

- We use random_state so that the same set of test and train data can be reproduced when you need to rerun the model.
- You can use any value as long as you keep the same value when you need to get the same set of test data

Scikit-Learn – Model Building

```
# Train the model using  Linear Regression
from sklearn.linear_model import LinearRegression
regressor_linear = LinearRegression()
regressor_linear.fit(X_train, y_train)
```

- We use only the training data to build the model. Model. Input columns in the X-train are the predictors and y_train are the labels or targets that we want to predict.
- By convention we use capital letter X for the training data as it represents a set of random variables.

Quiz - 5

Why do we use capital letter X to represent the training data when we use lower case y for the target variable that we predict?

a) X represents a set of independent variables.

b) It is mandatory to use X as capital letter for building a model.

c) X represents a set of random variables.

d) Both a and c

e) Both b and c

Scikit-Learn – Cross Validation

- Running a ten-fold cross validation

```
from sklearn.model_selection import cross_val_score

# Estimatingt the Cross Validation Score

cv_linear = cross_val_score(estimator = regressor_linear, X = X_train, y = y_train, cv = 10)

cv_linear
```

```
array([0.70126146, 0.80890564, 0.73921242, 0.72282719, 0.81675479,
       0.46358474, 0.76652097, 0.49539638, 0.75589085, 0.71450003])
```

Scikit-Learn – Predicting Target Results

```
# Predicting R2 Score for the Train set
from sklearn.metrics import r2_score

y_pred_linear_train = regressor_linear.predict(X_train)

r2_score_linear_train = r2_score(y_train, y_pred_linear_train)
r2_score_linear_train
```

```
0.7435787589010061
```

```
# Predicting R2 Score from the Test set results

y_pred_linear_test = regressor_linear.predict(X_test)
r2_score_linear_test = r2_score(y_test, y_pred_linear_test)
r2_score_linear_test
```

```
0.7133593313710349
```

Model Evaluation -Regression

```
# Calculating RMSE from the Test set results
from sklearn.metrics import mean_squared_error
rmse_linear = (np.sqrt(mean_squared_error(y_test, y_pred_linear_test)))

print("CV: ", cv_linear.mean())
print('R2_score (train): ', r2_score_linear_train)
print('R2_score (test): ', r2_score_linear_test)
print("RMSE: ", rmse_linear)
```

```
CV:  0.6984854476156042
R2_score (train):  0.7435787589010061
R2_score (test):  0.7133593313710349
RMSE:  4.647279745724213
```

Thank You