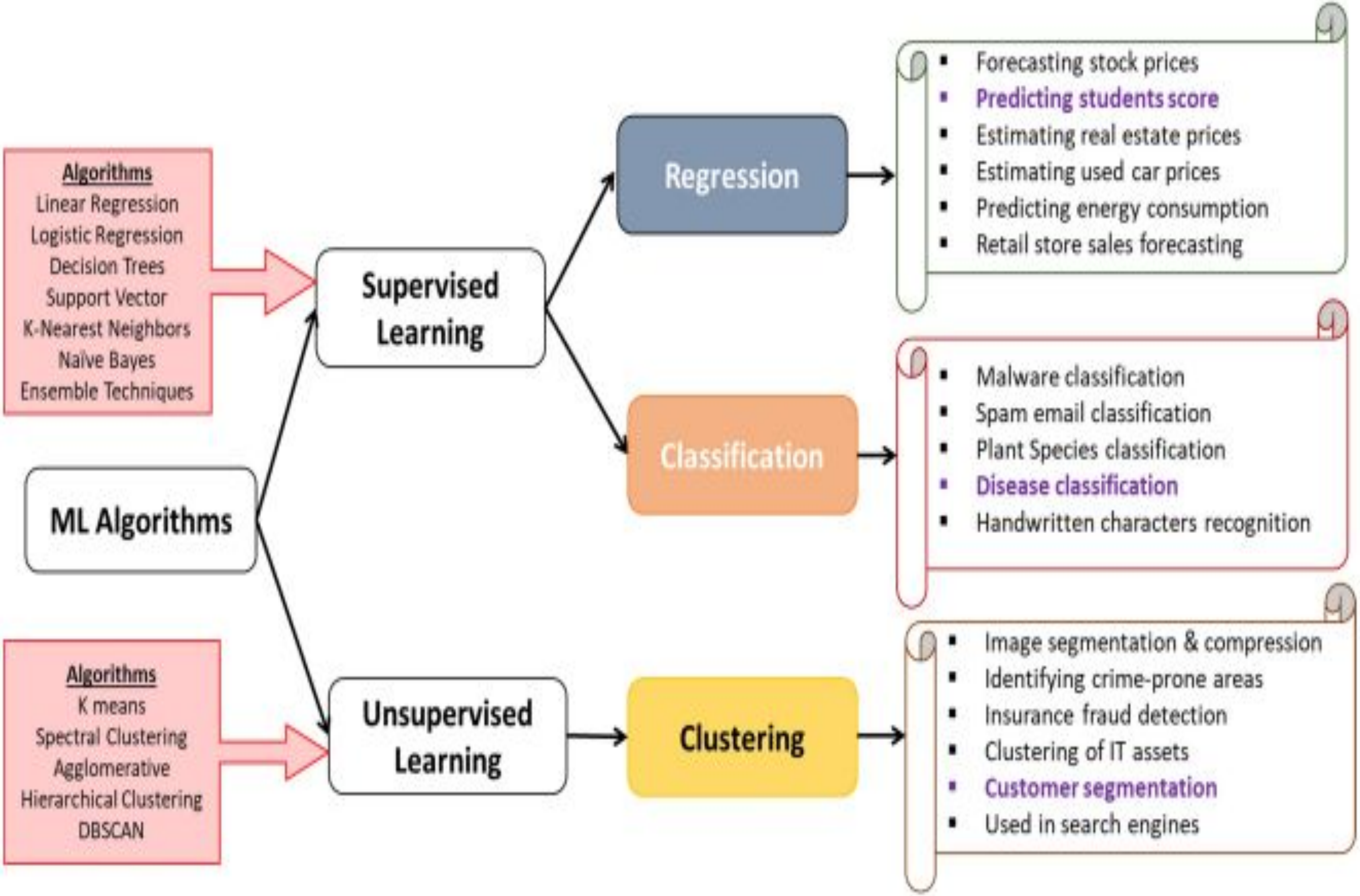
Applications



Algorithms
Linear Regression
Logistic Regression
Decision Trees
Support Vector
K-Nearest Neighbors
Naive Bayes
Ensemble Techniques

Supervised Learning

ML Algorithms

Algorithms
K means
Spectral Clustering
Agglomerative
Hierarchical Clustering
DBSCAN

Unsupervised Learning

Regression

- Forecasting stock prices
- Predicting students score
- Estimating real estate prices
- Estimating used car prices
- Predicting energy consumption
- Retail store sales forecasting

Classification

- Malware classification
- Spam email classification
- Plant Species classification
- Disease classification
- Handwritten characters recognition

Clustering

- Image segmentation & compression
- Identifying crime-prone areas
- Insurance fraud detection
- Clustering of IT assets
- Customer segmentation
- Used in search engines

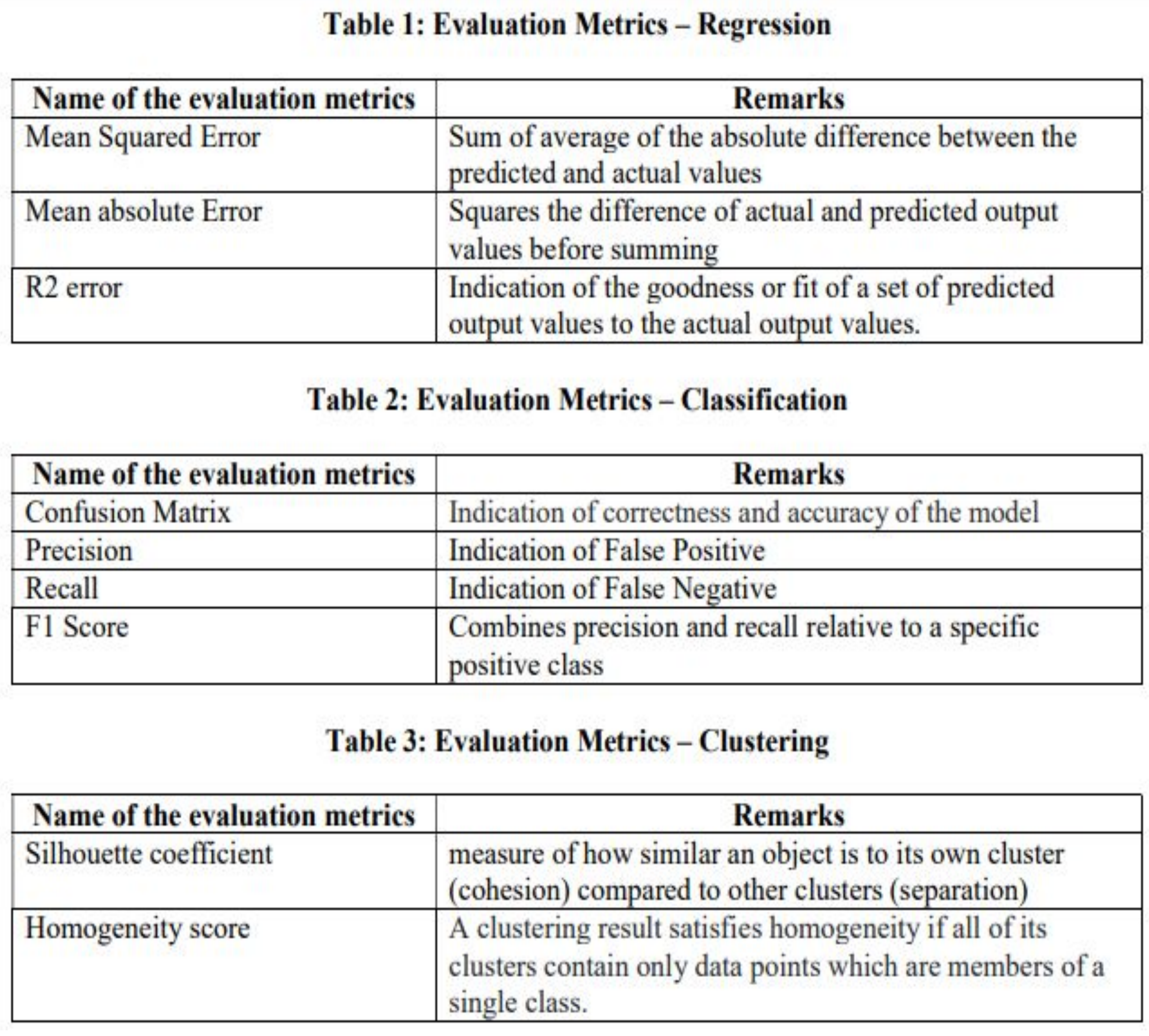


Table 1: Evaluation Metrics – Regression

Name of the evaluation metrics	Remarks
Mean Squared Error	Sum of average of the absolute difference between the predicted and actual values
Mean absolute Error	Squares the difference of actual and predicted output values before summing
R2 error	Indication of the goodness or fit of a set of predicted output values to the actual output values.

Table 2: Evaluation Metrics – Classification

Name of the evaluation metrics	Remarks
Confusion Matrix	Indication of correctness and accuracy of the model
Precision	Indication of False Positive
Recall	Indication of False Negative
F1 Score	Combines precision and recall relative to a specific positive class

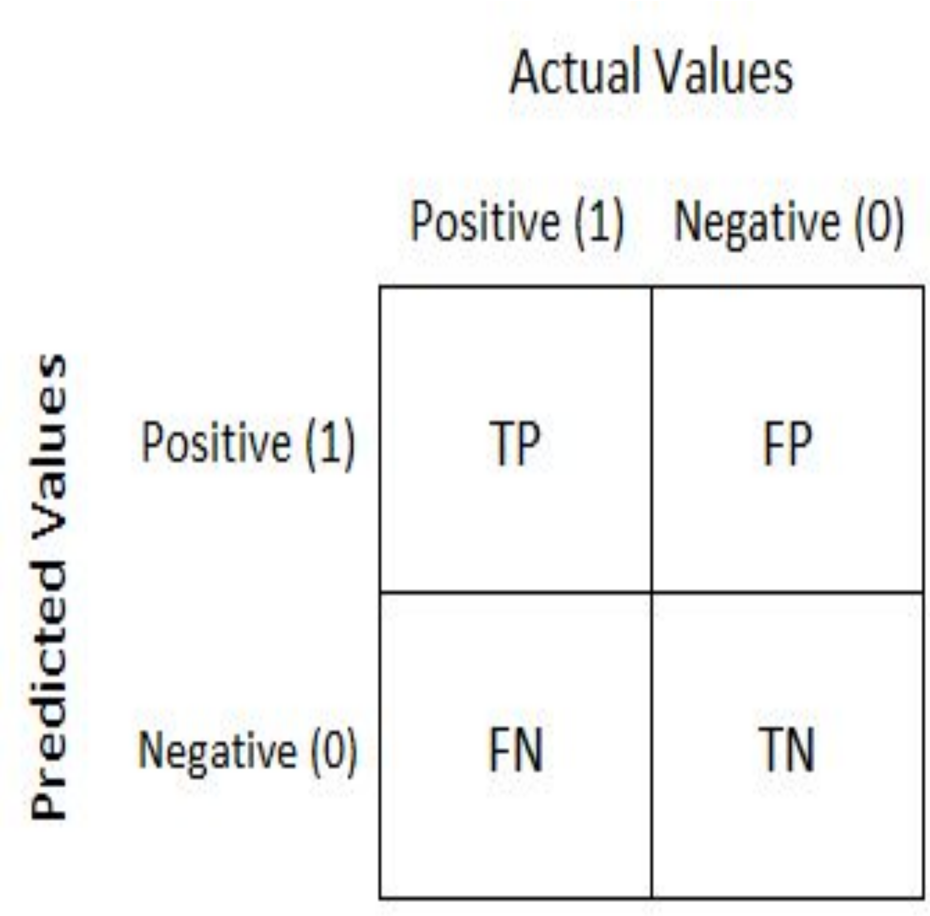
Table 3: Evaluation Metrics – Clustering

Name of the evaluation metrics	Remarks
Silhouette coefficient	measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation)
Homogeneity score	A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.

Description

- **True Positive** : These are cases in which we predicted yes (they have the disease), and they do have the disease
- **True negatives (TN):** We predicted no, and they don't have the disease
- **False positives (FP):** We predicted yes, but they don't actually have the disease
- **False negatives (FN):** We predicted no, but they actually do have the disease

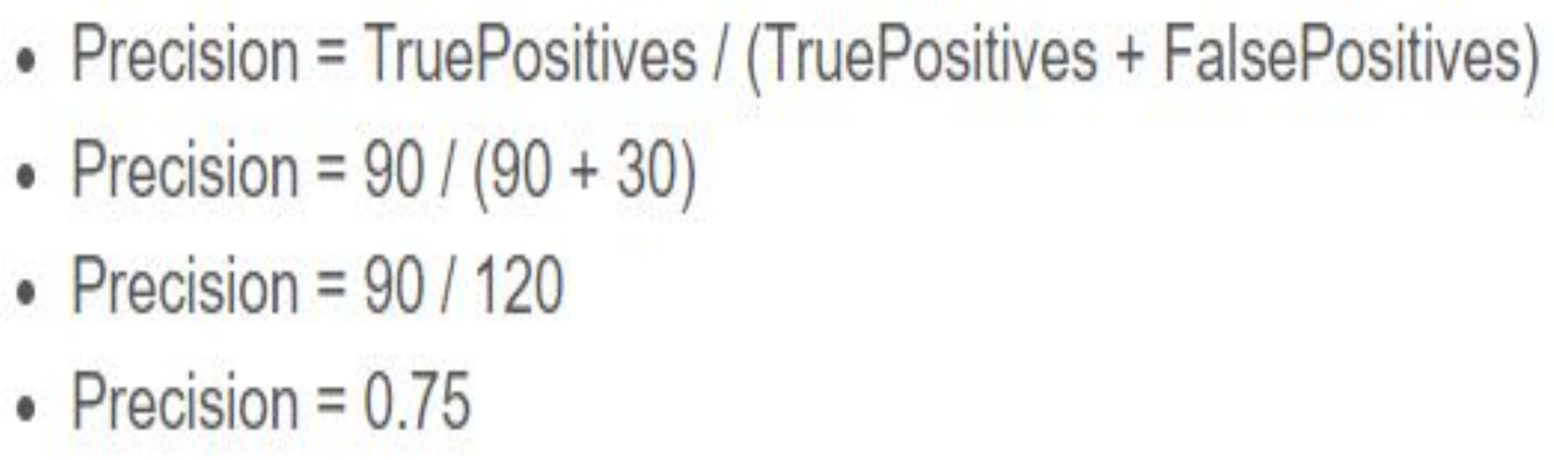
Confusion Matrix



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Precision

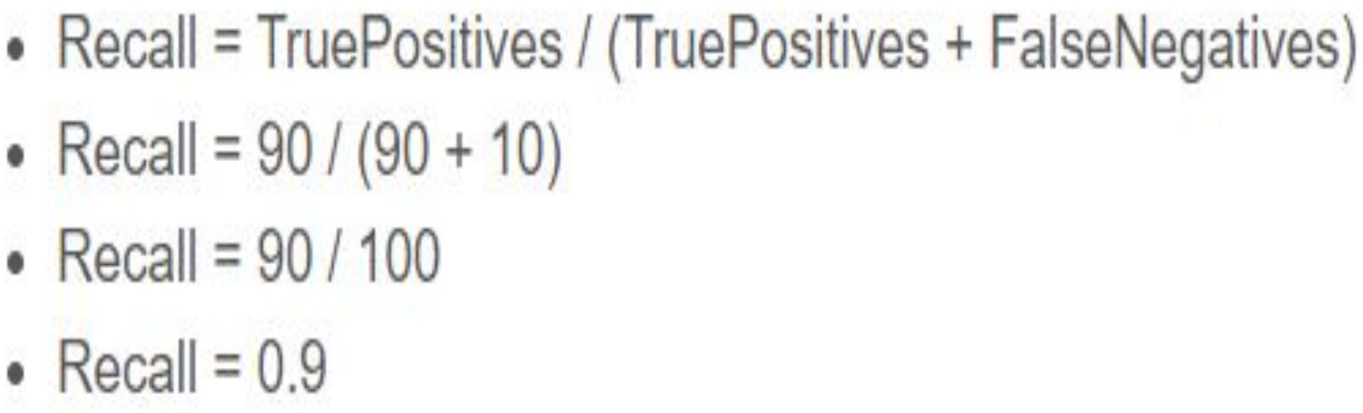
- A model makes predictions and predicts 120 examples, 90 of which are correct, and 30 of which are incorrect



- Precision = TruePositives / (TruePositives + FalsePositives)
- Precision = 90 / (90 + 30)
- Precision = 90 / 120
- Precision = 0.75

Recall

A model makes predictions and predicts 90 of the positive class predictions correctly and 10 incorrectly



- Recall = TruePositives / (TruePositives + FalseNegatives)
- Recall = 90 / (90 + 10)
- Recall = 90 / 100
- Recall = 0.9

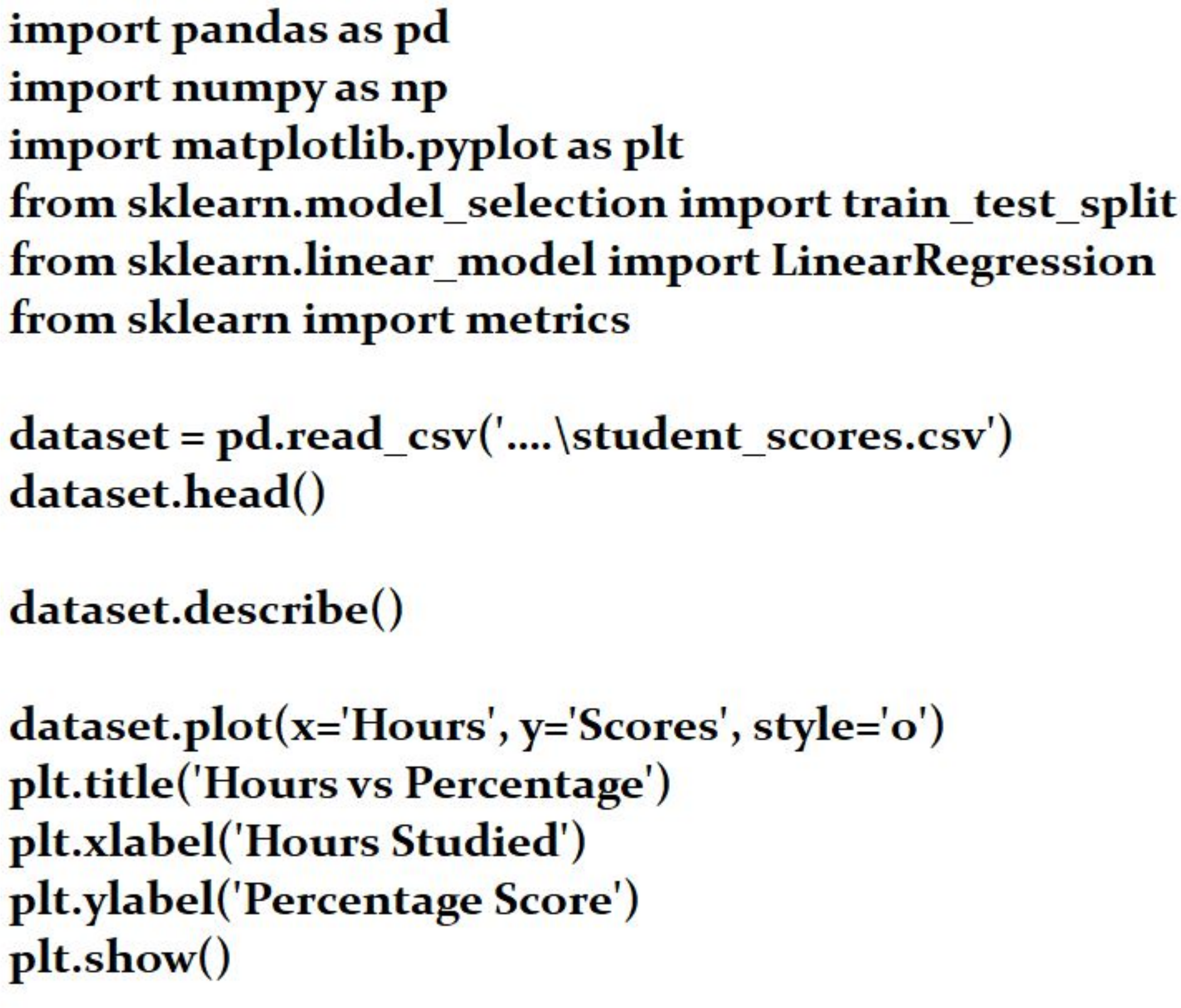
Steps

- Import required modules and packages
- Import data set Choose the right path for the dataset
- Descriptive statistics of the attributes available in the dataset
- Visualize the data
- Identify the independent (X) and dependent variables (y) in the data set
- Splitting the given data in to training set (80%) and testing set (20%)
- Model instantiation
- Model Training

Testing the model

Evaluation metrics

Example



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

dataset = pd.read_csv('....\student_scores.csv')
dataset.head()

dataset.describe()

dataset.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
print('X train shape: ', X_train.shape)
print('Y train shape: ', Y_train.shape)
print('X test shape: ', X_test.shape)
print('Y test shape: ', Y_test.shape)

regressor = LinearRegression()

regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(df)

print('Mean Absolute Error:',
metrics.mean_absolute_error (y_test, y_pred))
print('Mean Squared Error:',
metrics.mean_squared_error (y_test, y_pred))
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error (y_test, y_pred)))
```