**Lab Task 1 a: House price prediction**

**Dataset for this task is House\_price.csv**

**Step 1:** Import the relevant libraries such as numpy, pandas, train\_test\_split, Linear regression and metrics

**Step 2:** Read the House\_price.csv dataset given and check for null data. Since the task is to predict the house price, identify the response variable and explanatory variables.

**Step 3:** Convert any string data to numerical data using suitable conversion

**Step 4:** Split the data into train data and test data

**Step 5:** Train the Multiple linear regression model using training data. Note: You can use any ML algorithm as regressor.

**Step 6:** Predict the output for the test data using the fitted model

**Step 7:** Determine the performance of the model using suitable metric (List various metrics to analyze the performance of the regressor and write inference why you have chosen one that you are using in your code).

**Submit: Jupyter notebook file with comments for each cell of code and inference.**

**Lab Task 1 b: Simple Linear regression analysis**

**Dataset for this task is FuelConsumption.csv**

1. Go through the give notebook file and write code to perform the given tasks.
2. Write inference for each set of code you write to perform the given tasks.
3. Submitted the completed notebook file with inference notes.

Helping notes for reference:

**Pandas**

* In Python, the base library for data analysis is Pandas.
* It is a high-level library, built on the NumPy library, which is for scientific computing and numerical analysis.
* Pandas make it easier to work with data by offering its data structure, known as DataFrame.
* DataFrame helps in reading and storing your dataset. It provides the base functions for reading and writing the dataset

**Data Visualization**

* The two most used libraries for visualization are:
  + Matplotlib
  + Seaborn.
* Matplotlib is the base library for any visualizations in Python.
* Seaborn is also made on top of Matplotlib, which offers some of the most creative data visualization functions

**Matplotlib**

* To make the plotting interactive and view the same in your notebook itself, use

%matplotlib inline

* functions to control your plots:

Labels: xlabel(), ylabel(). They are for the x-axis and y-axis labels.

Legend: It is used for making the legend for the plot.

Title: To assign a title for your plot

Show function: To view the plot.

* Commonly used plots

plt.plot() – simple plot

plt.hist() – histogram

df.plot.bar() – Bar plot

plt.scatter(x\_axisdata, y\_axisdata) – Scatter plot

**Scikit** **learn**(**sklearn**)

* Scikit-learn is probably the most useful library for machine learning in Python.
* The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
* This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

**Components of scikit learn**

* **Supervised learning algorithms:** Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox.
* **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data using sklearn.
* **Unsupervised learning algorithms:** starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
* **Ensemble methods** − As name suggest, it is used for combining the predictions of multiple supervised models.
* **Various toy datasets:** Availability of various academic datasets (e.g. IRIS dataset, Boston House prices dataset).
* **Feature extraction:** Scikit-learn for extracting features from images and text (e.g. Bag of words)

**Commonly used Import commands for importing different libraries**

* **import** numpy **as** np
* **import** pandas **as** pd
* **import** matplotlib.pyplot **as** plt
* **import** seaborn **as** sns
* from sklearn.model\_selection import train\_test\_split
* from sklearn.metrics import r2\_score /use suitable metric

**Useful Linear Models in sklearn**

linear\_model.LinearRegression() /Linear regression

linear\_model.Ridge(alpha= ) /Ridge regression

linear\_model.Lasso(alpha=) /Lasso regression

linear\_model.ElasticNet() /Elasticnet regression

linear\_model.LogisticRegression() /Logistic regression

linear\_model.SGDClassifier() /Stochastic Gradient Descent classifier

linear\_model.SGDRegressor() /Stochastic Gradient Descent regressor

linear\_model.Perceptron() /perceptron model

**Supervised Learning Algorithms in sklearn**

**Support Vector Machines**

**Example**

**from** **sklearn** **import** svm

Support Vector Classification: **SVC**, **NuSVC** and **LinearSVC** are classes capable of performing binary and multi-class classification on a dataset.

Support Vector Regression: **SVR**, **NuSVR** and **LinearSVR**

**Naïve Bayes**

**Example**

**from** **sklearn.naive\_bayes** **import** GaussianNB

**GaussianNB** implements the Gaussian Naive Bayes algorithm for classification.

**MultinomialNB** implements the naive Bayes algorithm for multinomially distributed data

**ComplementNB** implements the complement naive Bayes (CNB) algorithm. CNB is an adaptation of the standard multinomial naive Bayes (MNB) algorithm that is particularly suited for imbalanced data sets.

[**BernoulliNB**](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB) implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions

[**CategoricalNB**](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.CategoricalNB.html#sklearn.naive_bayes.CategoricalNB) implements the categorical naive Bayes algorithm for categorically distributed data.

**Decision Trees**

**Example**

**from** **sklearn** **import** tree

clf = tree.DecisionTreeClassifier()

**DecisionTreeClassifier** is a class capable of performing multi-class classification on a dataset.

Decision trees can also be applied to regression problems, using the **DecisionTreeRegressor** class.

**Nearest Neighbors**

**sklearn.neighbors** provides functionality for unsupervised and supervised neighbors-based learning methods.

**Example**

**from** **sklearn.neighbors** **import** NearestNeighbors

**KNeighborsClassifier** implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user.

**KNeighborsRegressor** implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user.