Practical no.10

**Aim**: Implement Regression to predict fuel efficiency using Tensorflow (Auto MPG dataset).

# Theory

Regression is a method for understanding the relationship between independent variables or features and a dependent variable or outcome. Outcomes can then be predicted once the relationship between independent and dependent variables has been estimated.

# Material

* pandas
* tensorflow

# Program

import pandas as pd import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers # Load the Auto MPG dataset

url = ['h](http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data%27)t[tp://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'](http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data%27) column\_names = ['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight', 'Acceleration', 'Model Year', 'Origin']

raw\_dataset = pd.read\_csv(url, names=column\_names, na\_values='?', comment='\t',sep=' ', skipinitialspace=True)

dataset = raw\_dataset.copy()

dataset = dataset.dropna()

# Split the dataset into a training set and a test set train\_dataset = dataset.sample(frac=0.8, random\_state=0) test\_dataset = dataset.drop(train\_dataset.index)

# Separate the target variable, "MPG", from the input features train\_labels = train\_dataset.pop('MPG')

test\_labels = test\_dataset.pop('MPG') # Normalize the input features

train\_stats = train\_dataset.describe().transpose() def norm(x):

return (x - train\_stats['mean']) / train\_stats['std'] normed\_train\_data = norm(train\_dataset) normed\_test\_data = norm(test\_dataset)

# Define the model architecture model = keras.Sequential([

layers.Dense(64, activation='relu', input\_shape=[len(train\_dataset.keys())]), layers.Dense(64, activation='relu'),

layers.Dense(1)

])

# Compile the model

optimizer = tf.keras.optimizers.RMSprop(0.001) model.compile(loss='mse', optimizer=optimizer, metrics=['mae', 'mse']) # Train the model

EPOCHS = 1000

history = model.fit( normed\_train\_data, train\_labels,

epochs=EPOCHS, validation\_split = 0.2, verbose=0) # Evaluate the model on the test set

loss, mae, mse = model.evaluate(normed\_test\_data, test\_labels, verbose=2) print("Testing set Mean Abs Error: {:5.2f} MPG".format(mae))

# Make predictions on new data

test\_predictions = model.predict(normed\_test\_data).flatten() # Plot predicted vs actual values

import matplotlib.pyplot as plt plt.scatter(test\_labels, test\_predictions) plt.xlabel('True Values [MPG]') plt.ylabel('Predictions [MPG]') plt.axis('equal')

plt.axis('square') plt.xlim([0,plt.xlim()[1]])

plt.ylim([0,plt.ylim()[1]])

\_ = plt.plot([0, 100], [0, 100], color='red') plt.show()

**Output**

