Practical No :- 02

Aim: **Linear Regression**

a. Implement a simple linear regression model using TensorFlow's low-level API (or tf. keras).
b. Train the model on a toy dataset (e.g., housing prices vs. square footage).
c. Visualize the loss function and the learned linear relationship.

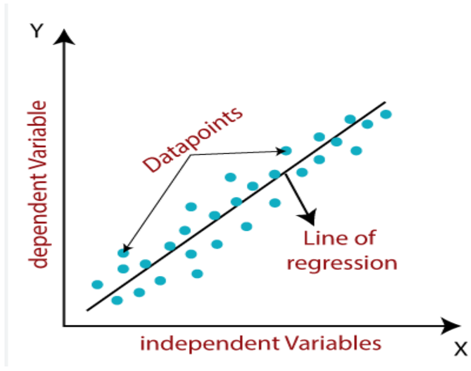
Make predictions on new data points.

Description:

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a straight line to the data. This line is represented by the equation y= ax+b where y is the dependent variable, x is the independent variable, a is the slope, and b is the y-intercept. It is used for prediction, analysis, and finding correlations between variables, and is a common technique in machine learning and data science.

Linear regression is a type of supervised machine-learning algorithm that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It assumes that there is a linear relationship between the input and output, meaning the output changes at a constant rate as the input changes. This relationship is represented by a straight line.
For example we want to predict a student's exam score based on how many hours they studied. We observe that as students study more hours, their scores go up. In the example of predicting exam scores based on hours studied. Here

- **Independent variable (input):** Hours studied because it's the factor we control or observe.
- **Dependent variable (output):** Exam score because it depends on hobw many hours were studied.



Code:

```
import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
```

a. Build a Simple Linear Regression Model

```
model = tf.keras.Sequential([
   tf.keras.layers.Dense(64, activation='relu', input_shape=(8,)),
   tf.keras.layers.Dense(64, activation='relu'),
   tf.keras.layers.Dense(1)
])
```

b. Train the Model on a toy dataset

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.california_housing.load_data()

# Scale features
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
          loss='mse',
          metrics=['mae'])

# Train the model
history = model.fit(x_train_scaled, y_train,
             validation_split=0.2,
             epochs=100,
             batch_size=32,
             verbose=1)
```

c. Visualize the Loss Curve and Learned Line

```
plt.figure(figsize=(8,5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('MSE Loss')
plt.legend()
plt.grid(True)
plt.title("Training and Validation Loss")
plt.show()
# Scatter plot: Actual vs Predicted
y_pred = model.predict(x_test_scaled).flatten()
plt.figure(figsize=(7,7))
plt.scatter(y_test, y_pred, alpha=0.5)
mins = min(min(y_test), min(y_pred))
maxs = max(max(y_test), max(y_pred))
plt.plot([mins, maxs], [mins, maxs], color='red', linestyle='--')
plt.xlabel('Actual Median House Value ($)')
plt.ylabel('Predicted Median House Value ($)')
plt.title('Actual vs Predicted House Values')
plt.grid(True)
plt.show()
```

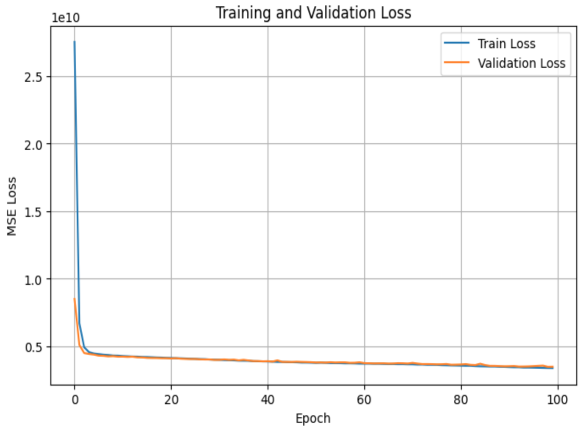
d. Make Predictions on New Data

```
# Use first 5 test samples as example
```

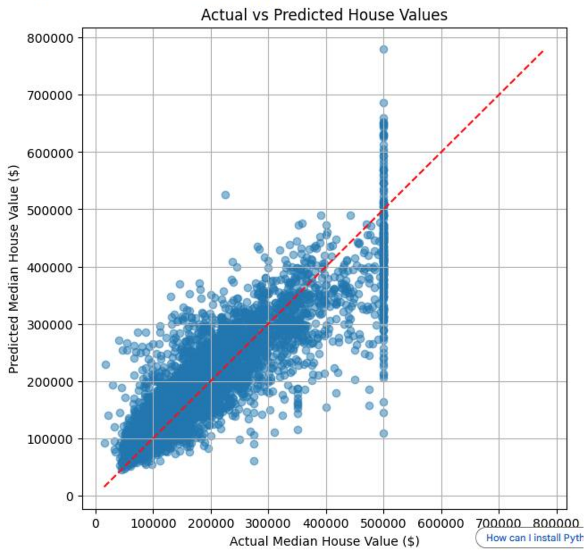
```
new_samples = x_test_scaled[:5]
predictions = model.predict(new_samples).flatten()
actuals = y_test[:5]

# Print nicely formatted output
print("House\tPredicted Price\t\tActual Price")
print("-"*45)
for i, (pred, actual) in enumerate(zip(predictions, actuals), start=1):
    print(f"{i}\t${pred:,.2f}\t\t${actual:,.2f}")
```

Output:



```
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step
House   Predicted Price     Actual Price
---------------------------------------------
1   $213,191.47     $397,900.00
2   $271,951.16     $227,900.00
3   $168,163.25     $172,100.00
4   $318,119.44     $186,500.00
5   $181,634.83     $148,900.00
```

Learnings:

In this practical, we implemented a **regression model** using TensorFlow/Keras to predict California housing prices based on multiple features. The model learned to approximate the relationship between input features and house prices by minimizing the **mean squared error** during training. Visualization of the **loss curve** confirmed that the model converged, while the **scatter plot of actual vs predicted values** showed how closely the predictions matched real prices. Using the trained model, we successfully predicted prices for new samples and compared them to actual values, demonstrating the model's predictive capability. This practical reinforced key concepts in **linear regression, feature scaling, model training, and evaluation**, and provided insight into applying deep learning techniques to real-world datasets.