Experiment # 9: Predict the price of a vehicle based on a set of characteristics using linear regression

Aim/Objective:

Program to Predict the price of a House based on a set of characteristics using linear regression
.

Description:

Description: Predicting the price of a vehicle based on a set of characteristics using linear regression involves building a model that learns the relationship between the features (characteristics) of a vehicle and their corresponding prices.

Pre-Requisites:

a. Basic understanding the prerequisites for implementing a vehicle price prediction system using linear regression involves foundational knowledge in several key areas.
b. Familiarity with programming concepts and basic knowledge of Python programminglanguage would be beneficial.

Pre-Lab:

1. Describe the key features (independent variables) that you think might influence vehicle prices. How would you categorize these features (numerical, categorical)?

- **Numerical:** Mileage, engine size, horsepower, age, fuel efficiency.
- **Categorical:** Brand, model type, fuel type, transmission, condition, previous owners.

2. Explain the terms "coefficients" and "intercept" in the context of linear regression. How are these values determined during model training?

- **Coefficients:** Measure how much the dependent variable (price) changes per unit change in a feature.
- **Intercept:** The predicted price when all features are zero.
- **Determination:** Found by minimizing prediction errors using methods like Ordinary Least Squares (OLS).

3. Define and explain the purpose of commonly used metrics such as Mean Squared Error (MSE) and R-squared (coefficient of determination). How do these metrics help evaluate the accuracy of a linear regression model?

- **MSE:** Measures average squared errors; lower is better.
- **R-squared:** Shows how well features explain price variations (closer to 1 = better fit).

4. What is feature engineering, and why is it important in the context of machine learning models like linear regression?

- Transforming raw data into meaningful features (handling missing values, encoding categories, scaling).
- Improves accuracy, reduces overfitting, and enhances model performance.

In-Lab:

Vehicle Predict the selling price of a vehicle based on a set of characteristics using ***linear regression.***

Data set: https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho/data

Procedure/Program:

```
from google.colab import files
uploaded = files.upload()

import pandas as pd

file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)

print("Dataset Preview:")
print(df.head())

print("\nColumn Names in Dataset:", df.columns)

df.columns = df.columns.str.lower()

df['age'] = 2025 - df['year']
df.drop(columns=['year', 'name'], inplace=True)

df = pd.get_dummies(df, drop_first=True)

print("\nDataset After Preprocessing:")
print(df.head())


X = df.drop(columns=['selling_price'])
y = df['selling_price']


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
print("\nModel Performance Metrics:")
print("R² Score:", r2_score(y_test, y_pred))
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))

import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.7, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
linestyle='dashed')
plt.xlabel("Actual Selling Price")
plt.ylabel("Predicted Selling Price")
plt.title("Actual vs Predicted Selling Price")
plt.show()
```

Data and Results:

Data

The dataset contains vehicle details like age, price, and specifications.

Result

Linear Regression predicts vehicle selling prices based on given attributes.

Analysis and Inferences:

Analysis

Model evaluation shows prediction accuracy using R^2, MAE, and MSE scores.

Inferences

Age, fuel type, and transmission significantly impact selling price predictions.

VIVA-VOCE Questions (In-Lab):

1. What is linear regression, and how is it used to predict car prices?

Linear regression models the relationship between car price (dependent variable) and features like age, mileage, and engine size. It predicts prices by fitting a line that minimizes errors.

2. What are the key features you consider when predicting car prices using linear regression?

- **Age & Mileage** (older/higher mileage cars cost less)
- **Engine size, brand, fuel type** (affect value)
- **Transmission type, condition, previous owners** (impact price)

3. Explain the Support Vector Machine (SVM) algorithm and its application in document classification.

SVM is a supervised learning algorithm that finds the optimal decision boundary for classification. In text classification, it converts documents into numerical features (TF-IDF, embeddings) and categorizes them effectively.

4. How do you preprocess data for training an SVM model for document classification?

- **Text cleaning** (remove stop words, punctuation)
- **Tokenization & Lemmatization** (reduce words to base form)
- **Feature extraction** (BoW, TF-IDF, embeddings)
- **Data splitting & scaling** (train/test, normalization)

5. What metrics do you use to evaluate the performance of your linear regression and SVM models?

- **Linear Regression:** MAE, MSE, R^2 score
- **SVM:** Accuracy, Precision, Recall, F1-score, Confusion Matrix

Post-Lab:

Train an SVM classifier on the Iris dataset and evaluate its performance using metrics like accuracy, precision, recall, and F1 score.

Dataset: Iris dataset.

Procedure/Program:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

iris = datasets.load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

svm_classifier = SVC(kernel='linear', random_state=42)
svm_classifier.fit(X_train, y_train)

y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')

print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
```

Data and Results:

Data

The Iris dataset contains 150 samples with four feature variables.

Result

The trained SVM model provides accuracy, precision, recall, and F1-score.

Analysis and Inferences:

Analysis

The SVM classifier performs well with standardized input feature values.

Inferences

SVM is effective for classifying Iris species based on features.

Evaluator Remark (if Any):	Marks Secured _____ out of 50
	Signature of the Evaluator with Date