Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 6

Aim: Regression and Its Types

- **a)** Implement simple linear regression using a dataset.
- **b)** Explore and interpret the regression model coefficients and goodness-of-fit measures.
- **c)** Extend the analysis to multiple linear regression and assess the impact of additional predictors.

CODE:

- ***Importing libraries***

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import mean_squared_error, r2_score
```

- ***Load Dataset***

```
df = pd.read_csv('insurance.csv')
print("Dataset Overview:\n")
display(df.head())
```

- ***Summary statistics***

```
print("\nSummary Statistics:\n")
display(df.describe())
```

- ***Check data types***

```
print("\nData Types:\n")
display(df.dtypes)
```

- ***Data Visualization***

```
# Distribution of target variable (charges)
plt.figure(figsize=(8,5))
sns.histplot(df['charges'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of Charges')
plt.xlabel('Charges')
plt.ylabel('Frequency')
plt.show()

# Relationship between age and charges
plt.figure(figsize=(8,5))
sns.scatterplot(x='age', y='charges', data=df, hue='smoker', style='sex', s=100)
plt.title('Age vs Charges')
plt.show()

# Pairplot for numerical variables
sns.pairplot(df, hue='smoker')
plt.show()
```

```
# Correlation heatmap
plt.figure(figsize=(8,6))
numeric_df = df.select_dtypes(include=np.number)
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap (Numeric Features)')
plt.show()
```

- ***Simple Linear Regression (SLR)***

```
# Split dataset into independent (X) and dependent (y) variables
X = df[['age']].values  # Age independent variable
y = df['charges'].values  # Charges dependent variable

# Split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train linear regression model
slr = LinearRegression()
slr.fit(X_train, y_train)

# Predict
y_train_pred = slr.predict(X_train)
y_test_pred = slr.predict(X_test)

# Model coefficients
print("Simple Linear Regression Coefficients:")
print(f"Slope (m): {slr.coef_[0]:.2f}")
print(f"Intercept (c): {slr.intercept_:.2f}")

# Plot training set vs predictions
plt.figure(figsize=(8,5))
plt.scatter(X_train, y_train, color='blue', label='Training Data')
plt.plot(X_train, y_train_pred, color='red', linewidth=2, label='Regression Line')
plt.title('SLR: Training Set - Age vs Charges')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.legend()
plt.show()

# Plot test set vs predictions
plt.figure(figsize=(8,5))
plt.scatter(X_test, y_test, color='green', label='Test Data')
plt.plot(X_test, y_test_pred, color='red', linewidth=2, label='Predictions')
plt.title('SLR: Test Set - Age vs Charges')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.legend()
plt.show()

# Performance metrics
print("SLR Performance on Test Set:")
print(f"R-squared: {r2_score(y_test, y_test_pred):.2f}")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_test_pred):.2f}")
```

- ***Multiple Linear Regression (MLR)***

```
# Independent variables: all except charges
```

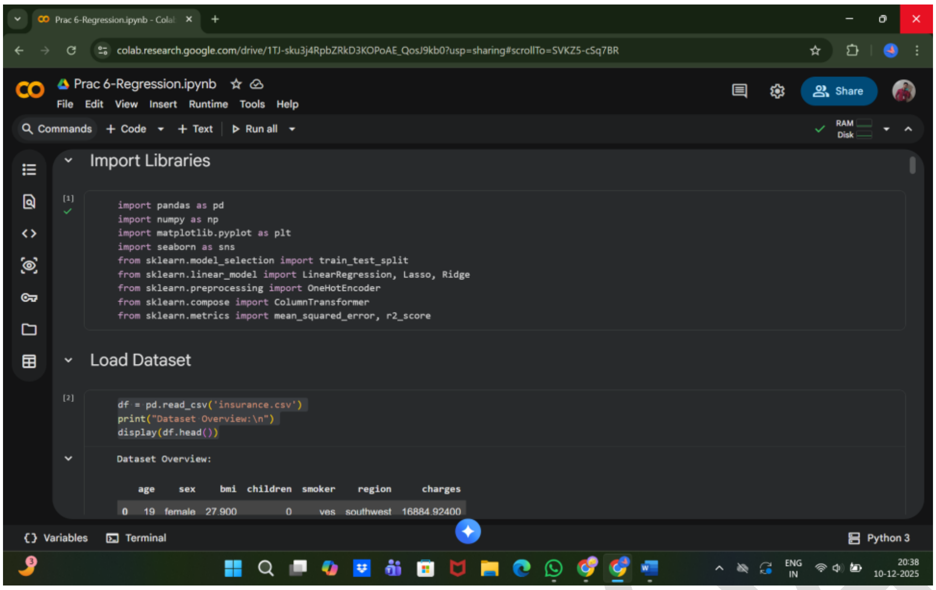
```
X = df.iloc[:, :-1].values
y = df['charges'].values

# One-hot encoding for categorical variables: 'sex', 'smoker', 'region'
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1,4,5])], remainder='passthrough')
X = np.array(ct.fit_transform(X))

# Split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Multiple Linear Regression model
mlr = LinearRegression()
mlr.fit(X_train, y_train)

# Predict
y_pred = mlr.predict(X_test)

# Performance metrics
print("Multiple Linear Regression Performance:")
print(f"R-squared: {r2_score(y_test, y_pred):.2f}")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred):.2f}")

# Plot Actual vs Predicted
plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, color='purple')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.title('MLR: Actual vs Predicted Charges')
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges')
plt.show()

# Lasso Regression (Regularization)
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
y_lasso_pred = lasso.predict(X_test)
print("Lasso Regression R-squared:", r2_score(y_test, y_lasso_pred))

# Ridge Regression (Regularization)
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_ridge_pred = ridge.predict(X_test)
print("Ridge Regression R-squared:", r2_score(y_test, y_ridge_pred))

# Elastic Net Regression
from sklearn.linear_model import ElasticNet
elastic = ElasticNet(alpha=0.1, l1_ratio=0.5)  # l1_ratio=0.5 balances L1 and L2
elastic.fit(X_train, y_train)
y_elastic_pred = elastic.predict(X_test)
print("Elastic Net R-squared:", r2_score(y_test, y_elastic_pred))
```

Output:

Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce

Data Science

PRACTICAL NO. 6



Import Libraries

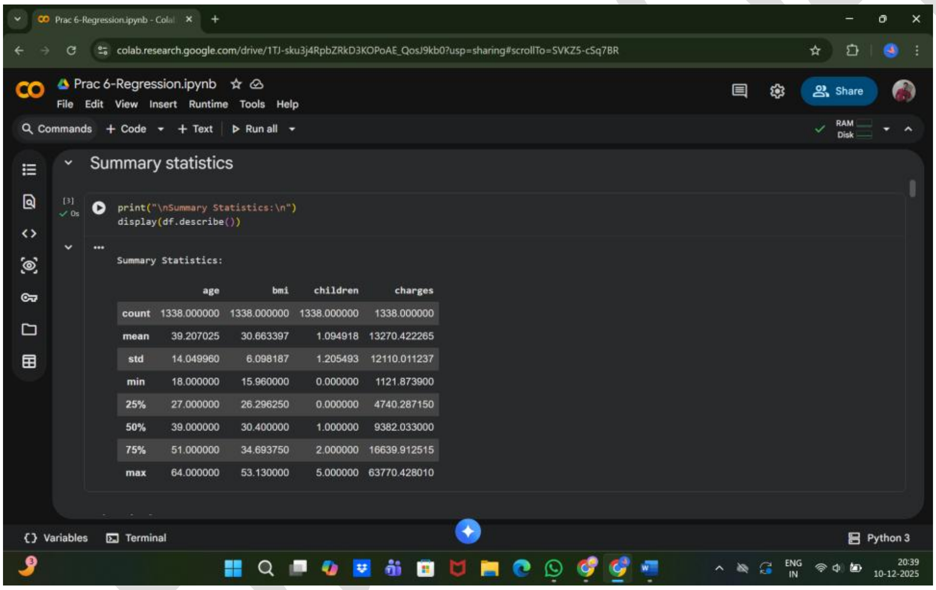
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import mean_squared_error, r2_score
```

Load Dataset

```python
df = pd.read_csv('insurance.csv')
print("Dataset Overview:\n")
display(df.head())
```

Dataset Overview:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400

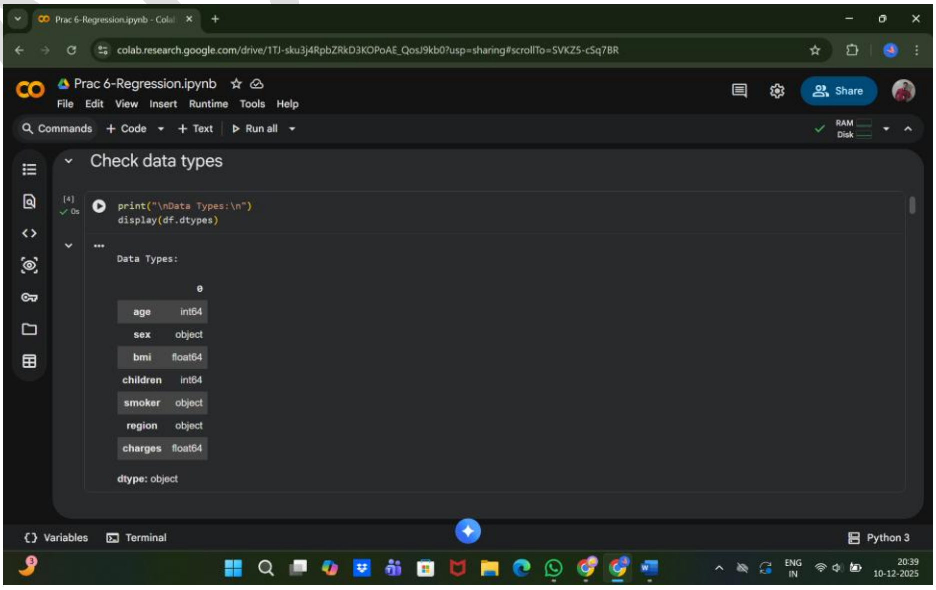


Summary statistics

```python
print("\nSummary Statistics:\n")
display(df.describe())
```

Summary Statistics:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010



Check data types

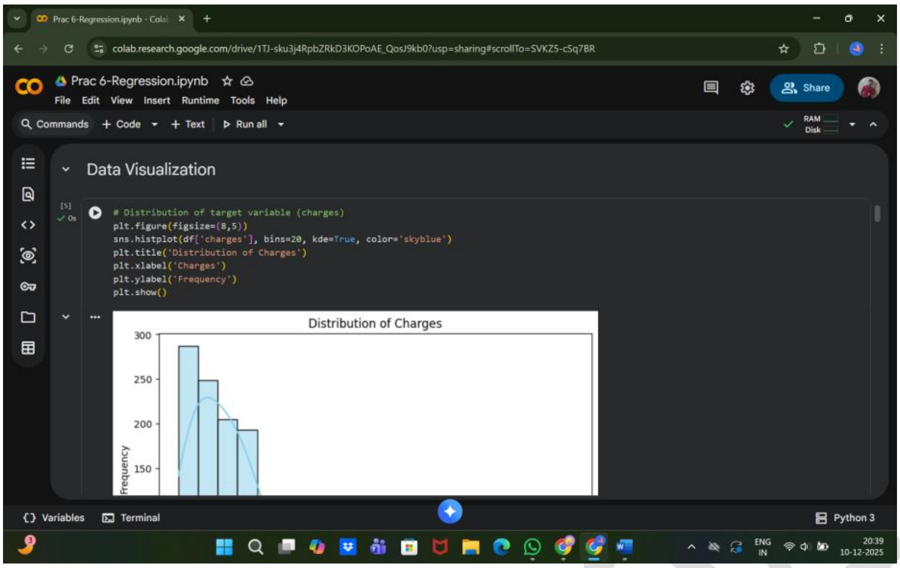
```python
print("\nData Types:\n")
display(df.dtypes)
```

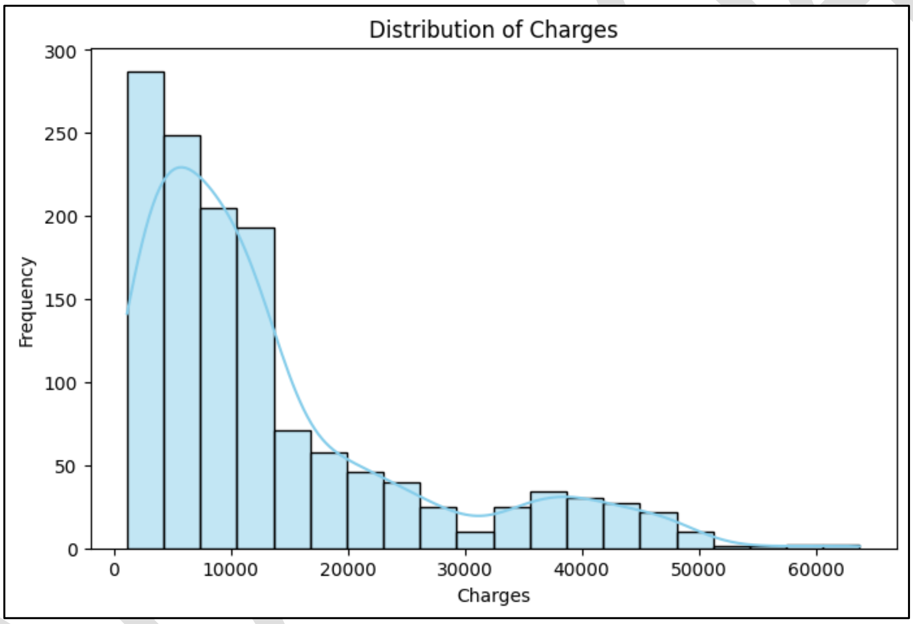
Data Types:

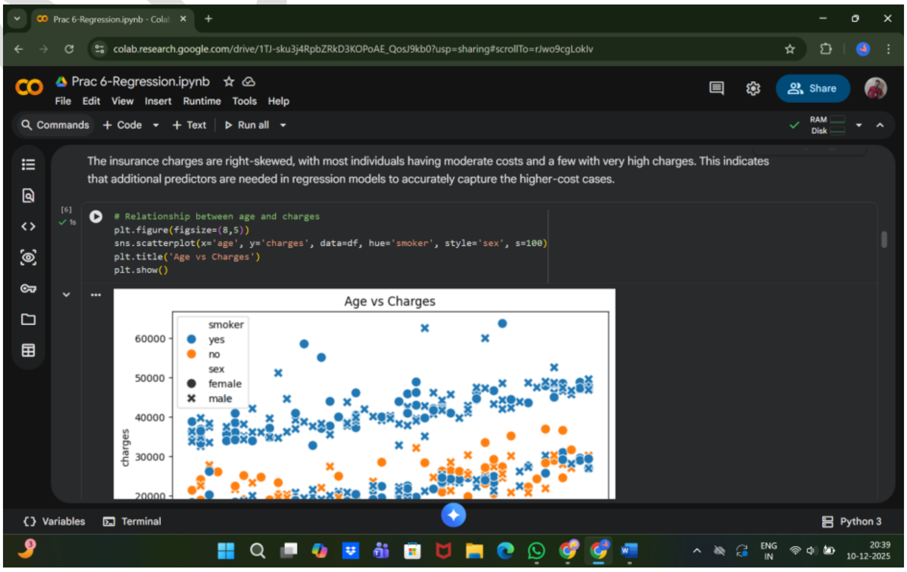
	0
age	int64
sex	object
bmi	float64
children	int64
smoker	object
region	object
charges	float64

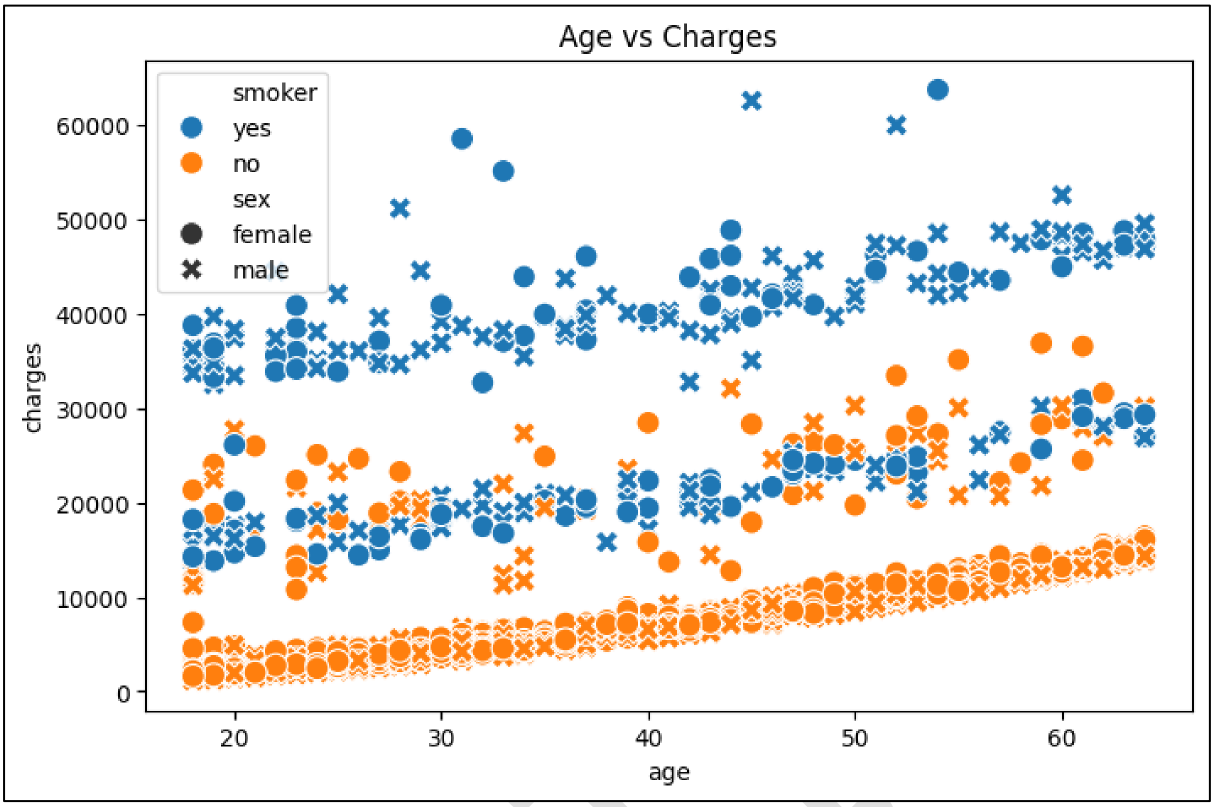
dtype: object

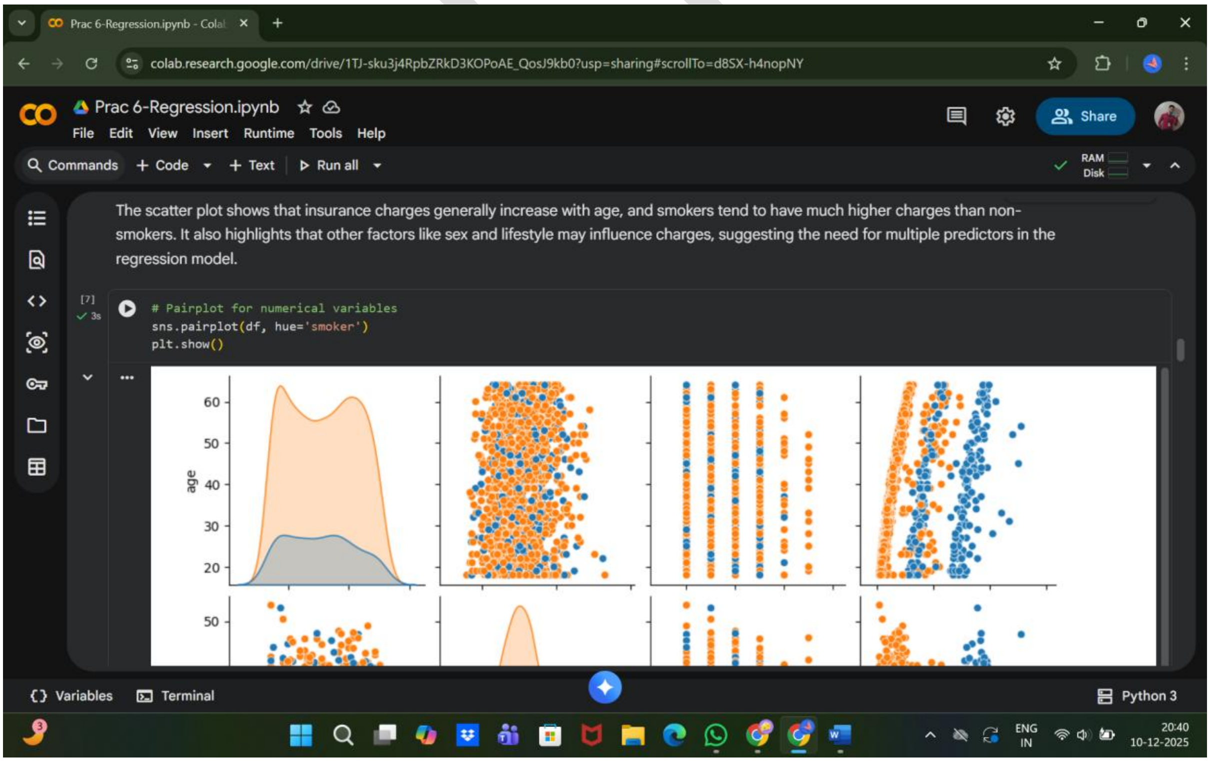
PRACTICAL NO. 6



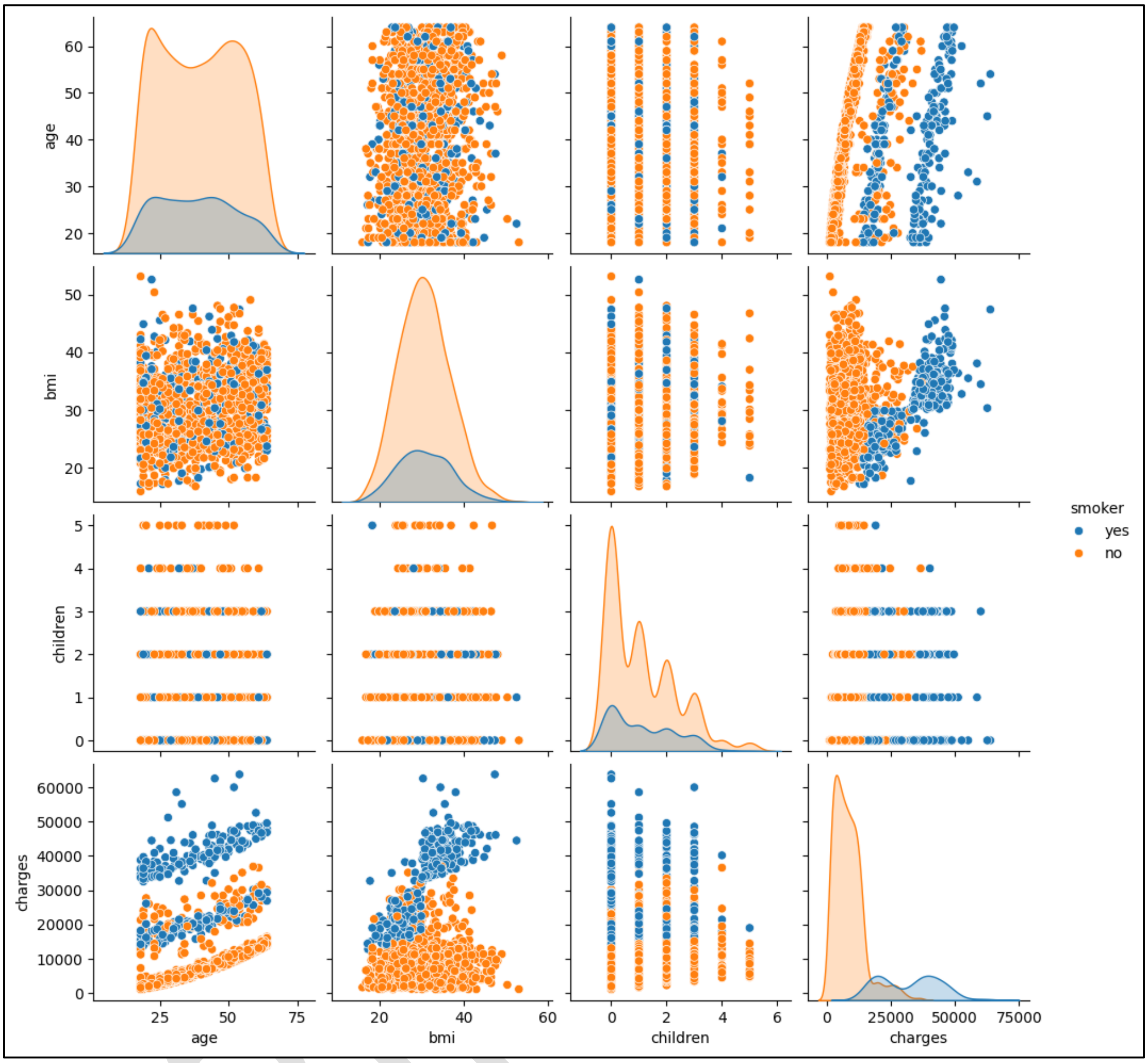


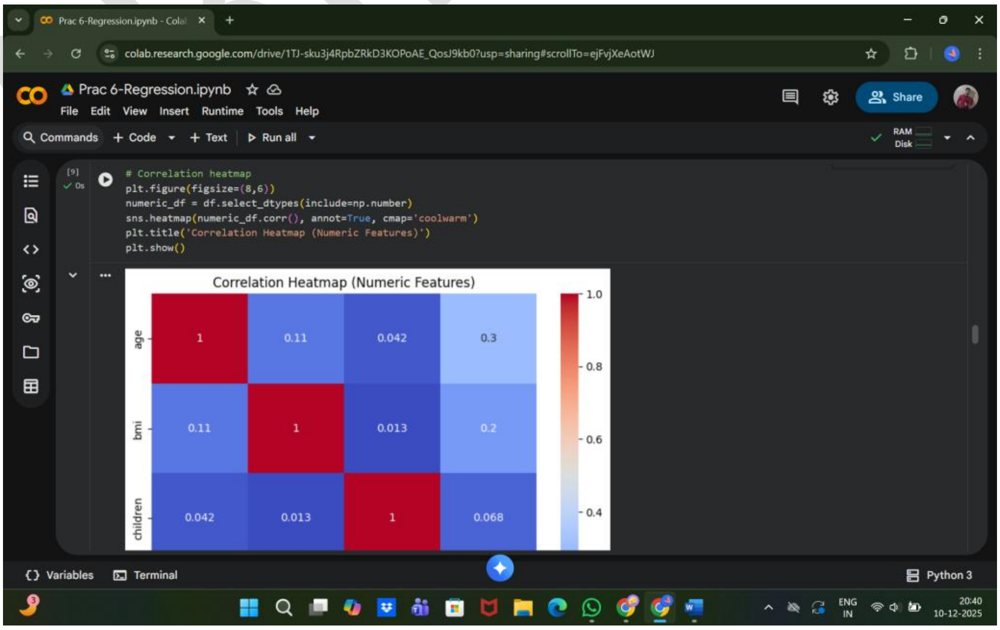


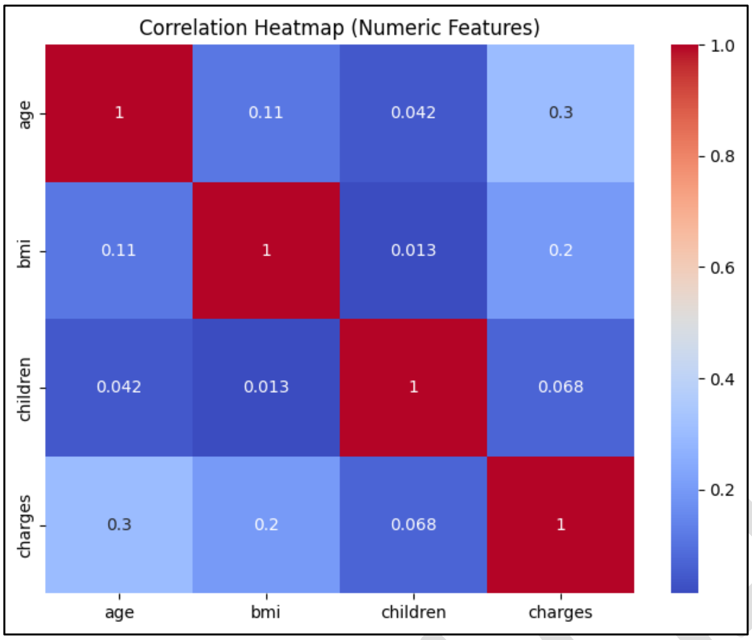

Age vs Charges
smoker
yes
no
sex
female
male
charges
age
0
10000
20000
30000
40000
50000
60000
20
30
40
50
60

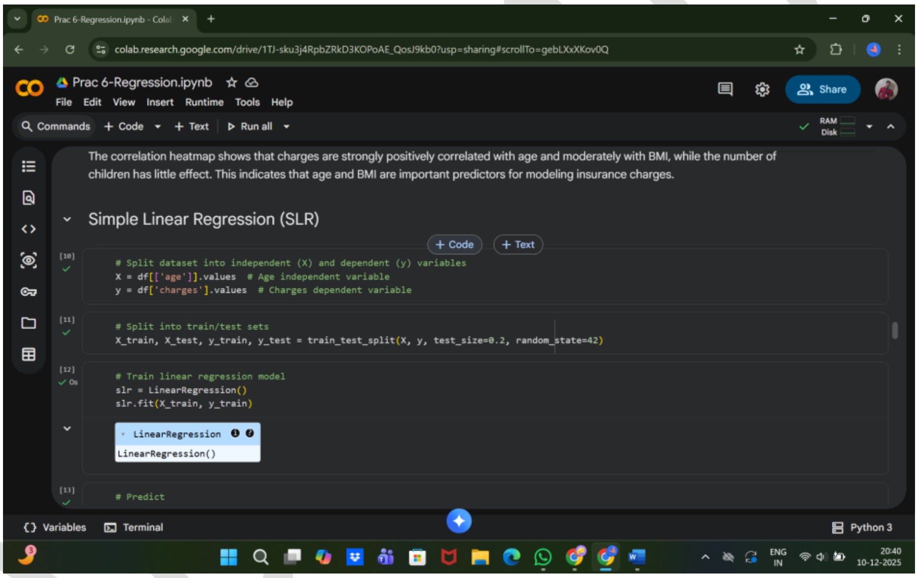

Prac 6-Regression.ipynb - Colab
colab.research.google.com/drive/1TJ-sku3j4RpbZRkD3KOPoAE_QosJ9kb0?usp=sharing#scrollTo=d8SX-h4nopNY
Prac 6-Regression.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
The scatter plot shows that insurance charges generally increase with age, and smokers tend to have much higher charges than non-smokers. It also highlights that other factors like sex and lifestyle may influence charges, suggesting the need for multiple predictors in the regression model.
# Pairplot for numerical variables
sns.pairplot(df, hue='smoker')
plt.show()
age
Variables
Terminal
Python 3
ENG
IN
20:40
10-12-2025

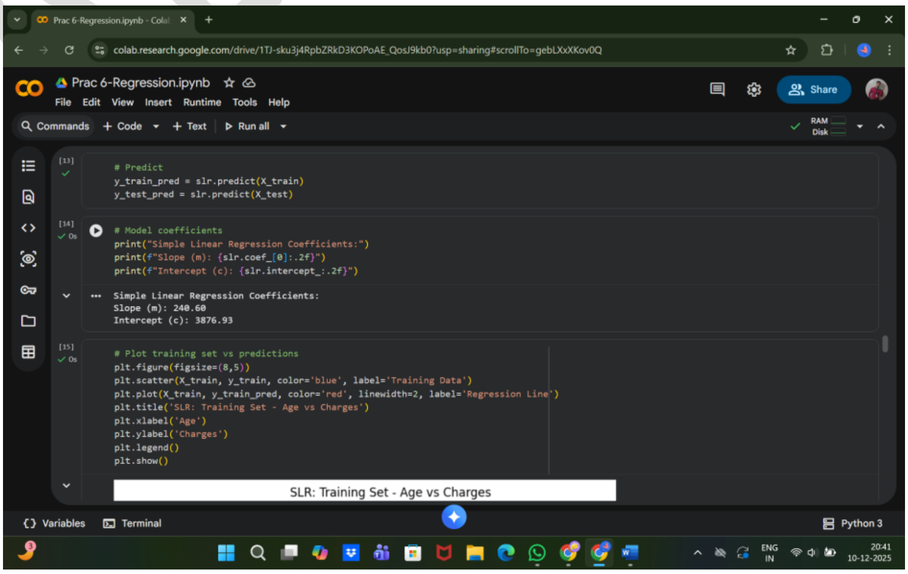
Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 6

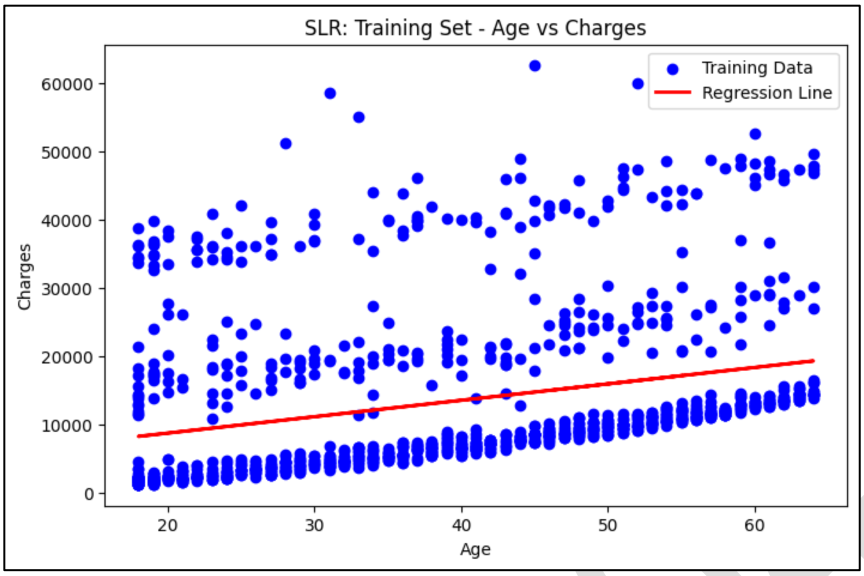


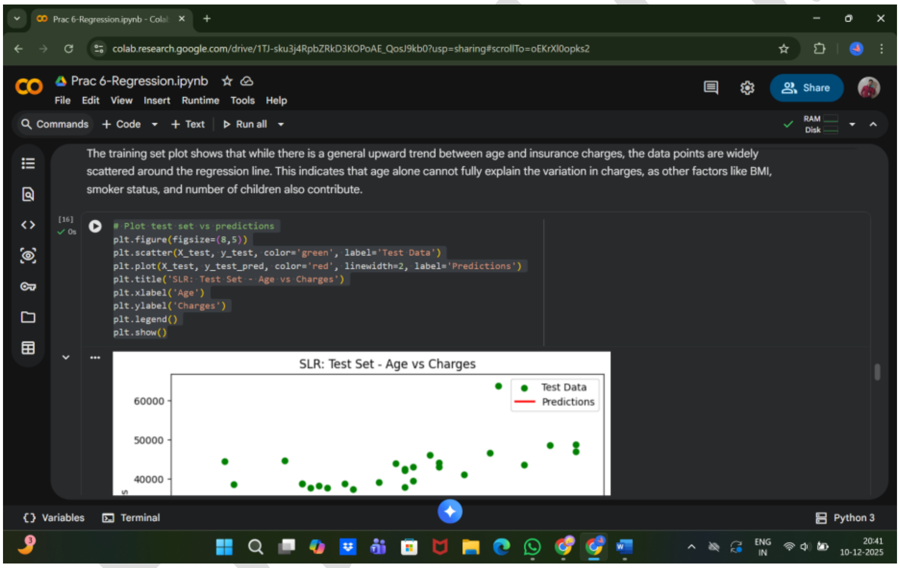


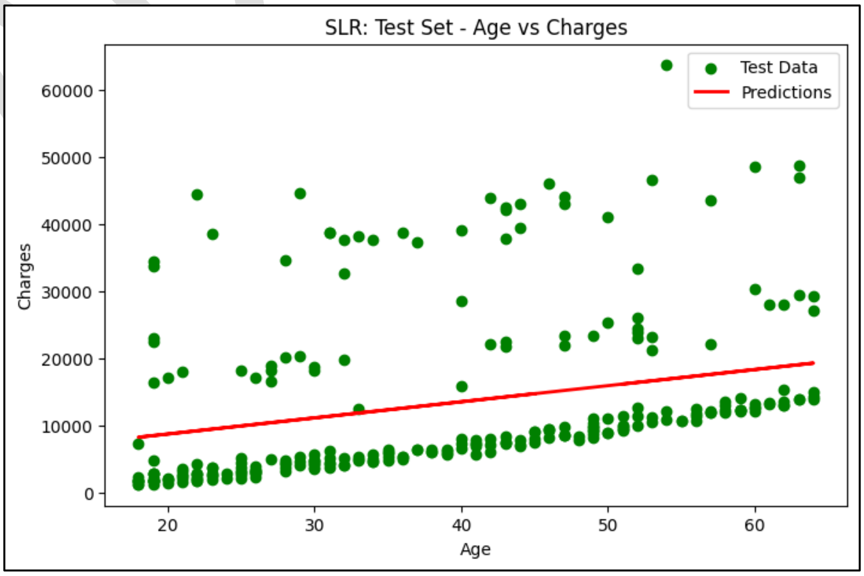

Correlation Heatmap (Numeric Features)
age
bmi
children
charges
1 0.11 0.042 0.3
0.11 1 0.013 0.2
0.042 0.013 1 0.068
0.3 0.2 0.068 1
1.0
0.8
0.6
0.4
0.2


Prac 6-Regression.ipynb
File Edit View Insert Runtime Tools Help
The correlation heatmap shows that charges are strongly positively correlated with age and moderately with BMI, while the number of children has little effect. This indicates that age and BMI are important predictors for modeling insurance charges.
Simple Linear Regression (SLR)
# Split dataset into independent (X) and dependent (y) variables
X = df[['age']].values # Age independent variable
y = df['charges'].values # Charges dependent variable
# Split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train linear regression model
slr = LinearRegression()
slr.fit(X_train, y_train)
LinearRegression
LinearRegression()
# Predict

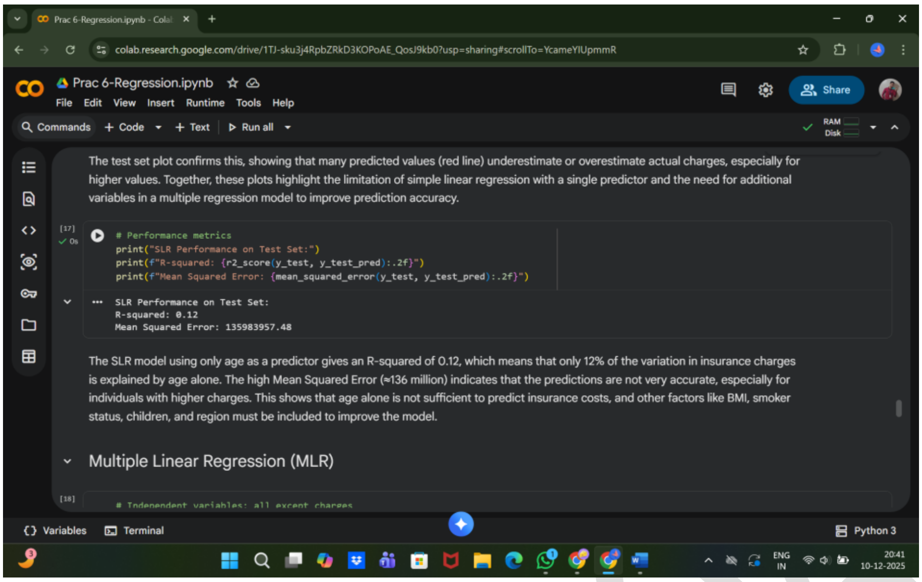

Prac 6-Regression.ipynb
File Edit View Insert Runtime Tools Help
# Predict
y_train_pred = slr.predict(X_train)
y_test_pred = slr.predict(X_test)
# Model coefficients
print("Simple Linear Regression Coefficients:")
print(f"Slope (m): {slr.coef_[0]:.2f}")
print(f"Intercept (c): {slr.intercept_:.2f}")
Simple Linear Regression Coefficients:
Slope (m): 240.60
Intercept (c): 3876.93
# Plot training set vs predictions
plt.figure(figsize=(8,5))
plt.scatter(X_train, y_train, color='blue', label='Training Data')
plt.plot(X_train, y_train_pred, color='red', linewidth=2, label='Regression Line')
plt.title('SLR: Training Set - Age vs Charges')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.legend()
plt.show()
SLR: Training Set - Age vs Charges


SLR: Training Set - Age vs Charges
Training Data
Regression Line
Charges
0
10000
20000
30000
40000
50000
60000
20
30
40
50
60
Age


Prac 6-Regression.ipynb - Colab
colab.research.google.com/drive/1TJ-sku3j4RpbZRkD3KOPoAE_QosJ9kb0?usp=sharing#scrollTo=oEKrXl0opks2
Prac 6-Regression.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
RAM
Disk
The training set plot shows that while there is a general upward trend between age and insurance charges, the data points are widely scattered around the regression line. This indicates that age alone cannot fully explain the variation in charges, as other factors like BMI, smoker status, and number of children also contribute.
[16]
# Plot test set vs predictions
plt.figure(figsize=(8,5))
plt.scatter(X_test, y_test, color='green', label='Test Data')
plt.plot(X_test, y_test_pred, color='red', linewidth=2, label='Predictions')
plt.title('SLR: Test Set - Age vs Charges')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.legend()
plt.show()
SLR: Test Set - Age vs Charges
Test Data
Predictions
60000
50000
40000
Variables
Terminal
Python 3


SLR: Test Set - Age vs Charges
Test Data
Predictions
Charges
0
10000
20000
30000
40000
50000
60000
20
30
40
50
60
Age

Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 6



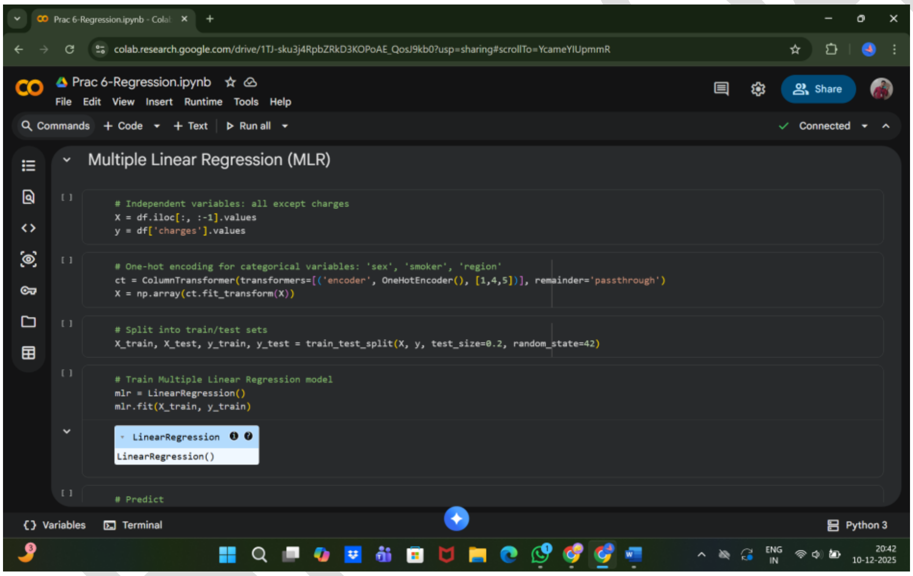
The test set plot confirms this, showing that many predicted values (red line) underestimate or overestimate actual charges, especially for higher values. Together, these plots highlight the limitation of simple linear regression with a single predictor and the need for additional variables in a multiple regression model to improve prediction accuracy.

```python
# Performance metrics
print("SLR Performance on Test Set:")
print(f"R-squared: {r2_score(y_test, y_test_pred):.2f}")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_test_pred):.2f}")
```

```
SLR Performance on Test Set:
R-squared: 0.12
Mean Squared Error: 135983957.48
```

The SLR model using only age as a predictor gives an R-squared of 0.12, which means that only 12% of the variation in insurance charges is explained by age alone. The high Mean Squared Error (≈136 million) indicates that the predictions are not very accurate, especially for individuals with higher charges. This shows that age alone is not sufficient to predict insurance costs, and other factors like BMI, smoker status, children, and region must be included to improve the model.

Multiple Linear Regression (MLR)



Multiple Linear Regression (MLR)

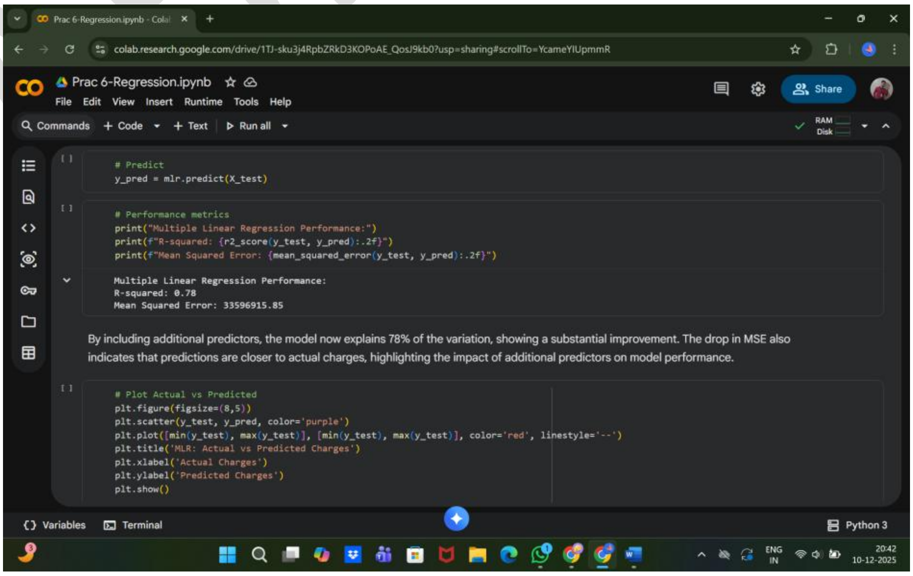
```python
# Independent variables: all except charges
X = df.iloc[:, :-1].values
y = df['charges'].values
```

```python
# One-hot encoding for categorical variables: 'sex', 'smoker', 'region'
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1,4,5])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

```python
# Split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
# Train Multiple Linear Regression model
mlr = LinearRegression()
mlr.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```



```python
# Predict
y_pred = mlr.predict(X_test)
```

```python
# Performance metrics
print("Multiple Linear Regression Performance:")
print(f"R-squared: {r2_score(y_test, y_pred):.2f}")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred):.2f}")
```

```
Multiple Linear Regression Performance:
R-squared: 0.78
Mean Squared Error: 33596915.85
```

By including additional predictors, the model now explains 78% of the variation, showing a substantial improvement. The drop in MSE also indicates that predictions are closer to actual charges, highlighting the impact of additional predictors on model performance.

```python
# Plot Actual vs Predicted
plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, color='purple')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.title('MLR: Actual vs Predicted Charges')
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges')
plt.show()
```

Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 6