Untitled | CODTECH_Task_2 | New Tab

localhost:8888/notebooks/CODTECH_Task_2.ipynb

jupyter CODTECH_Task_2 Last Checkpoint: 22 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

TASK TWO:PREDICTIVE MODELING WITH LINEAR REGRESSION Implement a simple linear regression model using a dataset with continuous target variables. Split the data into training and testing sets, train the model on the training data, evaluate its performance using metrics like mean squared error or R-squared, and make predictions on the test set. Visualize the regression line and actual vs. predicted values to assess the model's accuracy

[20]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

[21]:
```
# Step 1: Load the dataset
df = pd.read_csv('C:\\Users\\aadil\\Downloads\\covid_19_in_India.csv')
```

[22]:
```
#Step 2: Initial Data Inspection

# Display the first few rows
print(df.head())
```

```
        Date Name of State / UT  Latitude  Longitude  Total Confirmed cases  \
0  1/30/2020             Kerala   10.8505    76.2711                      1
1  1/31/2020             Kerala   10.8505    76.2711                      1
2   2/1/2020             Kerala   10.8505    76.2711                      2
3   2/2/2020             Kerala   10.8505    76.2711                      3
4   2/3/2020             Kerala   10.8505    76.2711                      3

   Death  Cured/Discharged/Migrated  New cases  New deaths  New recovered
0      0                          0          0           0              0
1      0                          0          0           0              0
2      0                          0          1           0              0
3      0                          0          1           0              0
```

Untitled | CODTECH_Task_2 | New Tab

localhost:8888/notebooks/CODTECH_Task_2.ipynb

jupyter CODTECH_Task_2 Last Checkpoint: 22 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[23]:
```
# Step 3: Data Preprocessing
# Check for missing values
df.isnull().sum()
```

[23]:
```
Date                        0
Name of State / UT          0
Latitude                    0
Longitude                   0
Total Confirmed cases       0
Death                       0
Cured/Discharged/Migrated   0
New cases                   0
New deaths                  0
New recovered               0
dtype: int64
```

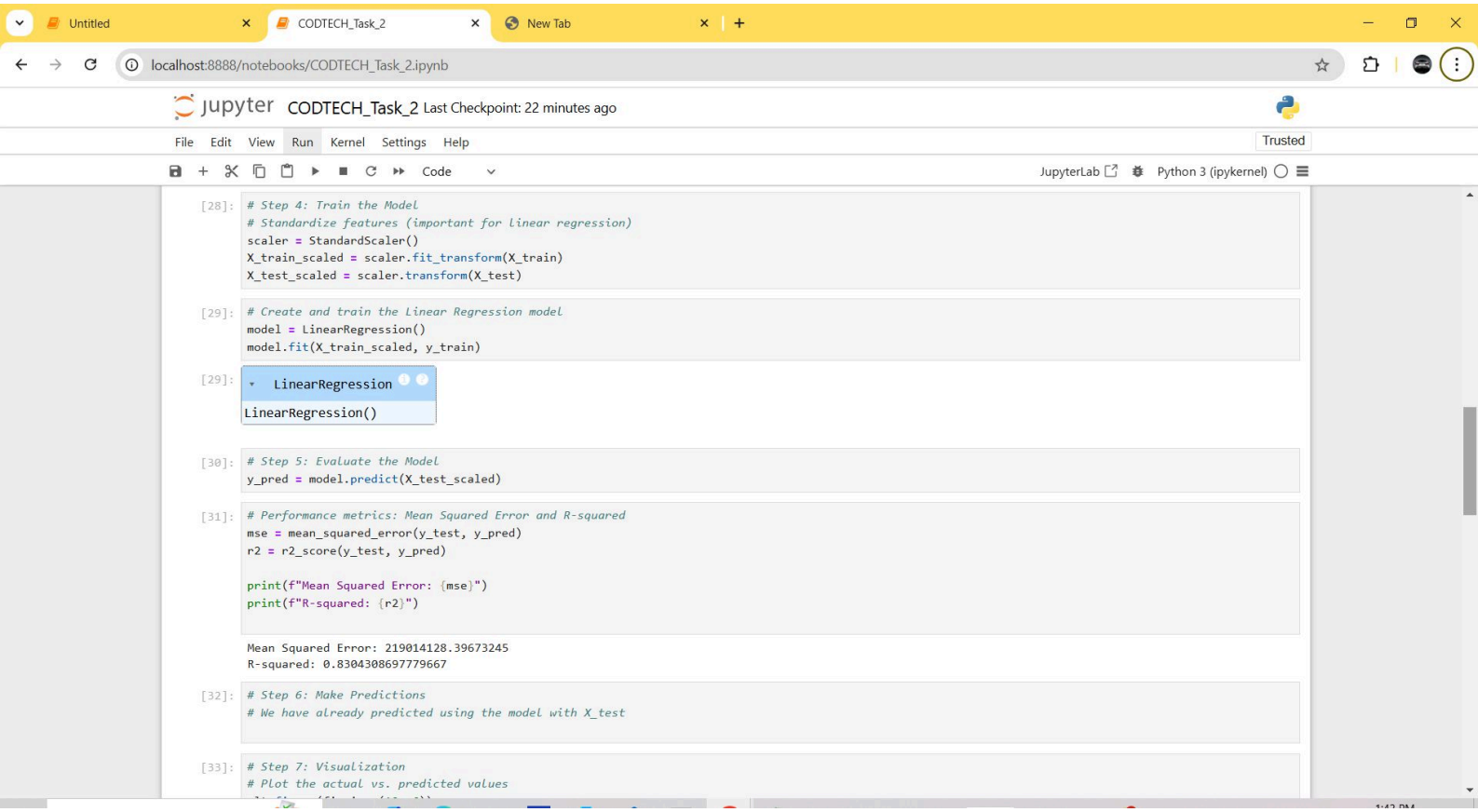
[24]:
```
# Let's drop any rows with missing target variable ('Total Confirmed cases') or fill them if necessary
df.dropna(subset=['Total Confirmed cases'], inplace=True)
```

[25]:
```
# We'll use 'Total Confirmed cases' as the target variable and 'New cases', 'Deaths', 'New deaths' as features
# Convert categorical variables to numeric if necessary (e.g., 'Name of State / UT')
df['State'] = df['Name of State / UT'].astype('category').cat.codes
```

[26]:
```
# Selecting Features and Target Variable
X = df[['Latitude', 'Longitude', 'New cases', 'New deaths', 'New recovered', 'State']]
y = df['Total Confirmed cases']
```

[27]:
```
# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[28]:
```
# Step 4: Train the Model
# Standardize features (important for linear regression)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```


Untitled
CODTECH_Task_2
New Tab
localhost:8888/notebooks/CODTECH_Task_2.ipynb
jupyter CODTECH_Task_2 Last Checkpoint: 22 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[28]:
# Step 4: Train the Model
# Standardize features (important for linear regression)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
[29]:
# Create and train the Linear Regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)
[29]:
LinearRegression
LinearRegression()
[30]:
# Step 5: Evaluate the Model
y_pred = model.predict(X_test_scaled)
[31]:
# Performance metrics: Mean Squared Error and R-squared
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
Mean Squared Error: 219014128.39673245
R-squared: 0.8304308697779667
[32]:
Step 6: Make Predictions
We have already predicted using the model with X_test
[33]:
Step 7: Visualization
Plot the actual vs. predicted values

Untitled | CODTECH_Task_2 | New Tab

localhost:8888/notebooks/CODTECH_Task_2.ipynb

jupyter CODTECH_Task_2 Last Checkpoint: 22 minutes ago

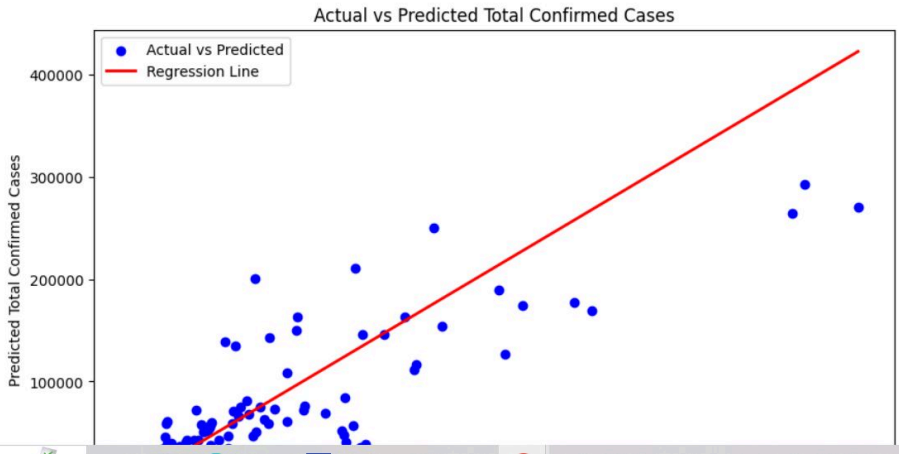
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[33]:
```
# Step 7: Visualization
# Plot the actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Actual vs Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2, label='Regression Line')
plt.xlabel('Actual Total Confirmed Cases')
plt.ylabel('Predicted Total Confirmed Cases')
plt.title('Actual vs Predicted Total Confirmed Cases')
plt.legend()
plt.show()
```



Untitled | CODTECH_Task_2 | New Tab

localhost:8888/notebooks/CODTECH_Task_2.ipynb

jupyter CODTECH_Task_2 Last Checkpoint: 22 minutes ago

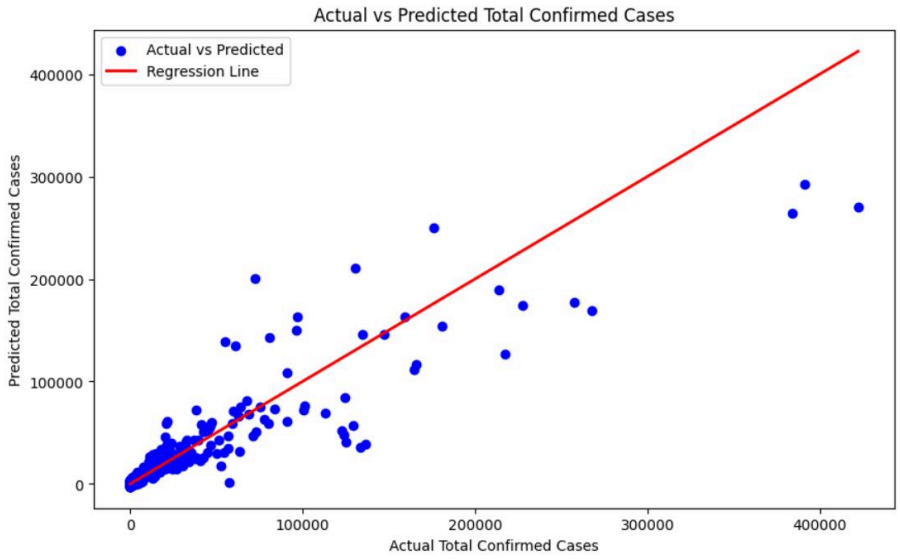
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

```
plt.show()
```



[34]:
```
# You can also plot a regression line if you are working with a single feature for simplicity
plt.scatter(X_test_scaled[:, 0], y_test, color='blue', label='Actual data')
```

Untitled | CODTECH_Task_2 | New Tab

localhost:8888/notebooks/CODTECH_Task_2.ipynb

jupyter CODTECH_Task_2 Last Checkpoint: 22 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[34]:
```
# You can also plot a regression line if you are working with a single feature for simplicity
plt.scatter(X_test_scaled[:, 0], y_test, color='blue', label='Actual data')
plt.plot(X_test_scaled[:, 0], model.predict(X_test_scaled), color='red', label='Regression Line')
plt.xlabel('Feature: Latitude')
plt.ylabel('Target: Total Confirmed Cases')
plt.title('Regression Line on a Single Feature')
plt.legend()
plt.show()
```