```
#lab2 linear regression code 1
# -*- coding: utf-8 -*-
"""Linear-Regression-Housing_Area_Price.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1CAlZml-P6V2V1RIrodgMfF8L3Ux
"""

import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt

df = pd.read_csv('housing_area_price.csv')
df

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='red',marker='+')

new_df = df.drop('price',axis='columns')
new_df

price = df.price
price

# Create linear regression object
reg = linear_model.LinearRegression()
reg.fit(new_df,price)

"""(1) Predict price of a home with area = 3300 sqr ft"""

reg.predict([[3300]])

reg.coef_

reg.intercept_

"""Y = m * X + b (m is coefficient and b is intercept)"""

3300*135.78767123 + 180616.43835616432

"""(1) Predict price of a home with area = 5000 sqr ft"""

reg.predict([[5000]])
```

File "<ipython-input-12-85b114609003 •••

Please explain the error:

```
  File "<ipython-input-12-85b114609003>",
    from sklearn.model_selection import tı
    ^
IndentationError: unexpected indent
```

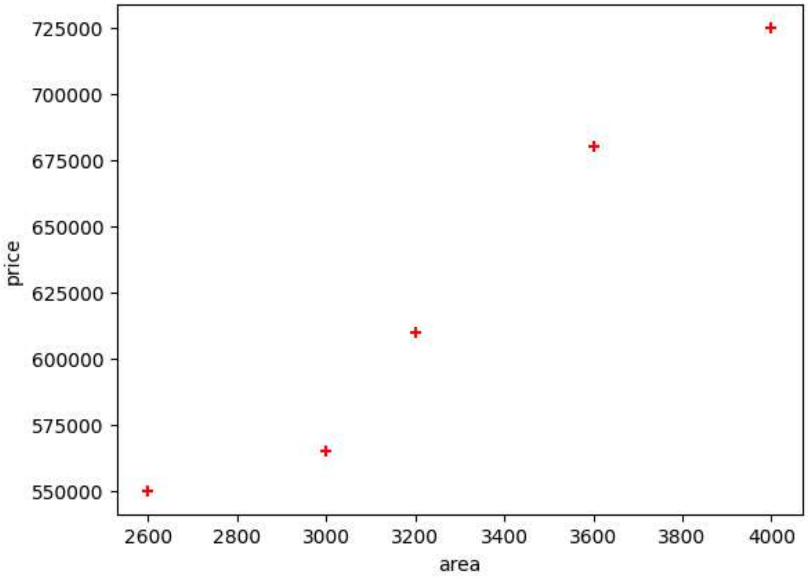
Hypothesis

Follow link (ctrl + click)

The error message `IndentationError: unexpected indent` indicates that there is an issue with the indentation in the code. Specifically, it points to the `from sklearn.model_selection import train_test_split` line. In Python, indentation is crucial for defining code blocks. This error typically occurs when a line or a block of code is indented more than it should be, or not indented when it should be.

Suggested Changes

```
#hiring
import pandas as pd
from sklearn.model_selection import trai
from sklearn.linear_model import LinearF
from sklearn.metrics import mean_squarec

# Step 1: Load the data
data = pd.read_csv('hiring.csv')

# Step 2: Clean the 'experience' column
# Replace 'eleven' with 11 (or appropria
data['experience'] = data['experience'].
# If there are other non-numeric values,
# You might need to use regular expressi

# If there are more than one string valu
def convert_to_int(val):
    word_to_num = {
        'one': 1,
        'two': 2,
        'three': 3,
        'four': 4,
        'five': 5,
        'six': 6,
        'seven': 7,
        'eight': 8,
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py
  warnings.warn(
array([859554.79452055])
```



```
#code 2
# -*- coding: utf-8 -*-
"""Multiple_LR_HomePrice.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1fK78C8TPV44HdvT6lsMhaau2wN
"""

import pandas as pd
import numpy as np
from sklearn import linear_model

df = pd.read_csv('homeprices_Multiple_LR.csv')
df

"""Data Preprocessing: Fill NA values with median value of a column"""

df.bedrooms.median()

df.bedrooms = df.bedrooms.fillna(df.bedrooms.median())
df

reg = linear_model.LinearRegression()
reg.fit(df.drop('price',axis='columns'),df.price)

reg.coef_

reg.intercept_

"""Find price of home with 3000 sqr ft area, 3 bedrooms, 40 year old"""
```

```
        'nine': 9,
        'ten': 10,
        'eleven': 11,
        'twelve': 12,
        'thirteen':13,
        'fourteen':14,
        'fifteen':15,
        'sixteen':16,
        'seventeen':17,
        'eighteen':18,
        'nineteen':19,
        'twenty':20,Follow link (ctrl + click)
    }
    try:
        # If it's already a number, retu
        return float(val)
    except ValueError:
        # If it's a word, convert it
        return word_to_num.get(val.lower


data['experience'] = data['experience'].

# Step 3: Define the features and target
X = data[['experience', 'test_score(out
y = data['salary($)']

# Step 4: Split the data into training a
X_train, X_test, y_train, y_test = train

# Step 5: Initialize the Linear Regressi
model = LinearRegression()
model.fit(X_train, y_train)

# Step 6: Check model accuracy
y_pred = model.predict(X_test)
print(f'Mean Squared Error: {mean_square

# Step 7: Make predictions for the new c
new_candidates = pd.DataFrame({
    'experience': [2, 12],
    'test_score(out of 10)': [9, 10],
    'interview_score(out of 10)': [6, 10
})

predicted_salaries = model.predict(new_c
print(f'Predicted Salaries: {predicted_s
```

Use code with caution

Explanation of changes:

- **Removed extra indentation:** The lines under the `import` statements were indented unnecessarily. This extra

```
reg.predict([[3000, 3, 40]])

112.06244194*3000 + 23388.88007794*3 + -3231.71790863*40 + 221323.0018
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.p
  warnings.warn(
498408.25157402386
```

```
#code 3 q1 canada
from google.colab import files
uploaded = files.upload()
```

Choose Files canada_per…income.csv

- **canada_per_capita_income.csv**(text/csv) - 874 bytes, last modified: 3/10/2025 - 100% done

```
#canada
import pandas as pd
import io
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression

df = pd.read_csv("canada.csv")
print(df.head())
missing_values = df.isnull().sum()
# Display columns with missing values
print(missing_values[missing_values > 0])

df.drop_duplicates(inplace = True)

plt.xlabel('year')
plt.ylabel('per capita income')
plt.scatter(df['year'], df['per capita income (US$)'], color='red', ma
plt.show()

X = df[['year']] #independent variable (predictor)
y = df['per capita income (US$)'] #dependent variable (target)

reg = LinearRegression()#req 2 parameters
reg.fit(X,y)
predicted_income = reg.predict([[2025]])
print(predicted_income)
```

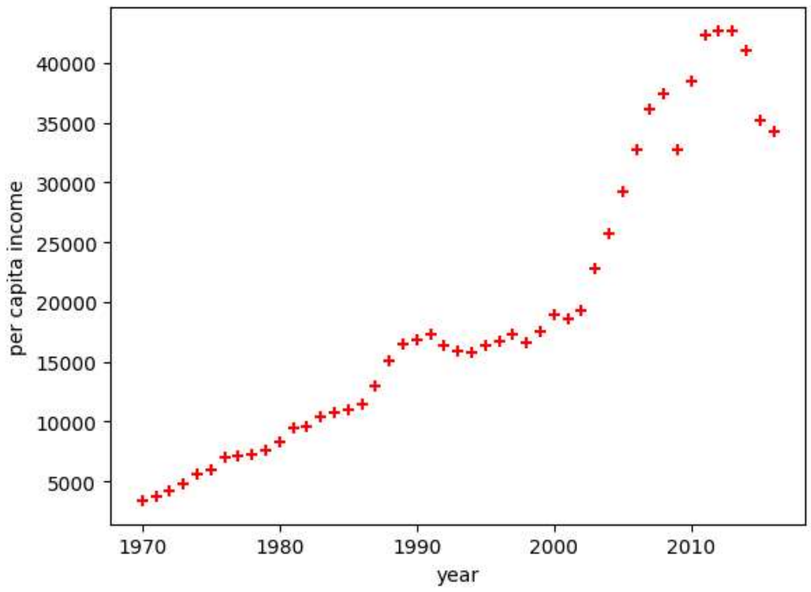
indentation caused the `IndentationError`. By removing the extra spaces at the beginning of these lines and bringing them to the same level as the `import` statements, the code now has correct indentation and should execute without the error.

▶ Sources

Rate this answer

Follow link (ctrl + click)

```
   year  per capita income (US$)
0  1970              3399.299037
1  1971              3768.297935
2  1972              4251.175484
3  1973              4804.463248
4  1974              5576.514583
Series([], dtype: int64)
```



```
[45431.01947053]
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py
  warnings.warn(
```

Follow link (ctrl + click)

```
#code 4
#ssalary
from google.colab import files
uploaded = files.upload()
```

Choose Files salary.csv

- **salary.csv**(text/csv) - 346 bytes, last modified: 3/10/2025 - 100% done

```
Saving salary.csv to salary (1).csv
```

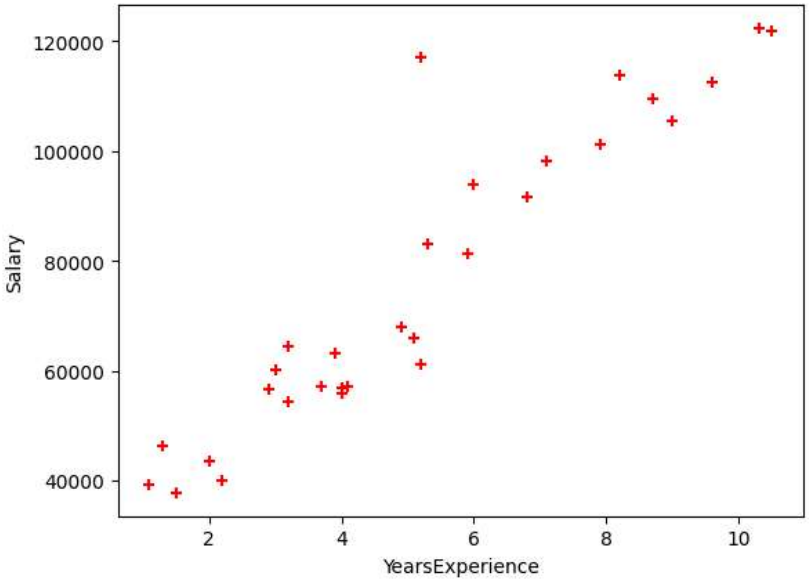
```
#salary
import pandas as pd
import io
from sklearn import linear_model
import numpy as np


df = pd.read_csv("salary.csv")
print(df.head())

df.replace(' ',np.nan,inplace = True)
print(df.head())

missing_values = df.isnull().sum()
# Display columns with missing values
print(missing_values[missing_values > 0])

#handle missing values
from sklearn.impute import SimpleImputer
```

```
imputer2 = SimpleImputer(strategy="mean")

df_copy=df

# Step 2: Fit the imputer on the "Age" and "Salary"column
# Note: SimpleImputer expects a 2D array, so we reshape the column

imputer2.fit(df_copy[["YearsExperience"]])

# Step 3: Transform (fill) the missing values in the "Age" and "Salary'

df_copy["YearsExperience"] = imputer2.transform(df[["YearsExperience"]]

# Verify that there are no missing values left

print(df_copy["YearsExperience"].isnull().sum())

plt.xlabel('YearsExperience')
plt.ylabel('Salary')
plt.scatter(df_copy['YearsExperience'], df_copy['Salary'], color='red',
plt.show()

X = df_copy[['YearsExperience']] #independent
y = df_copy['Salary'] #dependent

reg = linear_model.LinearRegression()
reg.fit(X,y)
predicted_salary = reg.predict([[12]])
print(predicted_salary)
```

Follow link (ctrl + click)

```
   YearsExperience  Salary
0              1.1   39343
1              1.3   46205
2              1.5   37731
3              2.0   43525
4              2.2   39891
   YearsExperience  Salary
0              1.1   39343
1              1.3   46205
2              1.5   37731
3              2.0   43525
4              2.2   39891
YearsExperience    2
dtype: int64
0
```

Follow link (ctrl + click)



```
[139980.88923969]
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.p
  warnings.warn(
```

```
#multiple linear regression
#code  5
from google.colab import files
uploaded = files.upload()
```

Choose Files hiring.csv

- **hiring.csv**(text/csv) - 198 bytes, last modified: 3/10/2025 - 100% done

```
Saving hiring.csv to hiring (1).csv
```

```
#hiring
import pandas as pd
import io
from sklearn import linear_model
import numpy as np
from sklearn.preprocessing import OrdinalEncoder
from sklearn.impute import SimpleImputer

df = pd.read_csv("hiring.csv")
print(df.head())

df.replace(' ',np.nan,inplace = True)
```

```
print(df.head())

missing_values = df.isnull().sum()
# Display columns with missing values
print(missing_values[missing_values > 0])


df['experience'].fillna("unknown", inplace=True)
print(df.head())



#handle missing values
ordinal_encoder = OrdinalEncoder(categories=[["unknown","one", "two","
# Fit and transform the data
df['experience_encoded'] = ordinal_encoder.fit_transform(df[['experien

print(df.head())

df.drop('experience',axis = 1,inplace = True)
print(df.head())

from sklearn.impute import SimpleImputer
imputer2 = SimpleImputer(strategy="mean")

df_copy=df

# Step 2: Fit the imputer on the "Age" and "Salary"column
# Note: SimpleImputer expects a 2D array, so we reshape the column

imputer2.fit(df_copy[["test_score(out of 10)"]])

# Step 3: Transform (fill) the missing values in the "Age" and "Salary

df_copy["test_score(out of 10)"] = imputer2.transform(df[["test_score(

# Verify that there are no missing values left

print(df_copy["test_score(out of 10)"].isnull().sum())

X = df_copy[['test_score(out of 10)','interview_score(out of 10)','exp
y = df_copy[['salary($)']]

reg = linear_model.LinearRegression()
reg.fit(X,y)
predicted_salary = reg.predict([[2,9,6]])
print(predicted_salary)

predicted_salary = reg.predict([[12,10,10]])
print(predicted_salary)
```

Follow link (ctrl + click)

```
  experience  test_score(out of 10)  interview_score(out of 10)
0        NaN                    8.0                           9
1        NaN                    8.0                           6
2       five                    6.0                           7
3        two                   10.0                          10
4      seven                    9.0                           6
  experience  test_score(out of 10)  interview_score(out of 10)
0        NaN                    8.0                           9
1        NaN                    8.0                           6
2       five                    6.0                           7
3        two                   10.0                          10
4      seven                    9.0                           6
experience               2
test_score(out of 10)    1
```

```
dtype: int64
  experience  test_score(out of 10)  interview_score(out of 10)
0    unknown                    8.0                           9
1    unknown                    8.0                           6
2       five                    6.0                           7
3        two                   10.0                          10
4      seven                    9.0                           6
  experience  test_score(out of 10)  interview_score(out of 10)
0    unknown                    8.0                           9
1    unknown                    8.0                           6
2       five                    6.0                           7
3        two                   10.0                          10
4      seven                    9.0                           6

   experience_encoded
0                 0.0
1                 0.0
2                 5.0
3                 2.0
4                 7.0
   test_score(out of 10)  interview_score(out of 10)  salary($)
0                    8.0                           9      50000
1                    8.0                           6      45000
2                    6.0                           7      60000
3                   10.0                          10      65000
4                    9.0                           6      70000

   experience_encoded
0                 0.0
1                 0.0
2                 5.0
3                 2.0
4                 7.0
0
[[57801.7884606]]
[[90438.68025262]]
<ipython-input-15-885684c19c67>:20: FutureWarning: A value is tr
The behavior will change in pandas 3.0. This inplace method will

For example, when doing 'df[col].method(value, inplace=True)', t


  df['experience'].fillna("unknown", inplace=True)
```

Follow link (ctrl + click)

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

Choose Files 1000_Companies.csv

- **1000_Companies.csv**(text/csv) - 52203 bytes, last modified: 3/10/2025 - 100% done

```
Saving 1000 Companies.csv to 1000 Companies (4).csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder

# Load company data
df_companies = pd.read_csv('1000_Companies.csv')

# Handle categorical variable (State)
label_encoder = LabelEncoder()
df_companies['State'] = label_encoder.fit_transform(df_companies['State'
# Define features and target variable
X_companies = df_companies[['R&D Spend', 'Administration', 'Marketing Sp
y_companies = df_companies['Profit']

# Handle missing values by filling with median
df_companies.fillna(df_companies.median(), inplace=True)

# Train the model
reg_companies = LinearRegression()
reg_companies.fit(X_companies, y_companies)
```

Enter a prompt here ⊕

0/2000

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more

Follow link (ctrl + click)