DESCRIPTION

were are currently updating our project and you will be regularly updated and for more infomation notes,pdf,code,full png ans svg map you can follow us on following and DM on social media platforms.

`instagram` https://www.instagram.com/ahanger.asif?igsh=MXZiczBreWgwd25iMA==

`facebook` https://www.facebook.com/ahangerasifahanger.asif

In [1]:
```
print("HI Asif ")
```

```
HI Asif
```

In [2]:
```
import numpy as np
```

In [6]:
```
x = [range(20)]

y = [np.sin(i) for i in x]
y
```

Out[6]:
```
[array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
        -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849,
        -0.54402111, -0.99999021, -0.53657292,  0.42016704,  0.99060736,
         0.65028784, -0.28790332, -0.96139749, -0.75098725,  0.14987721])]
```

In [3]:
```
import pandas as pd
import numpy as numpy
import matplotlib.pyplot as pyplot
import seaborn as sns

from sklearn.linear_model import LinearRegression
```

In [22]:
```
df =  sns.load_dataset('tips')
df.head()
```

Out[22]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

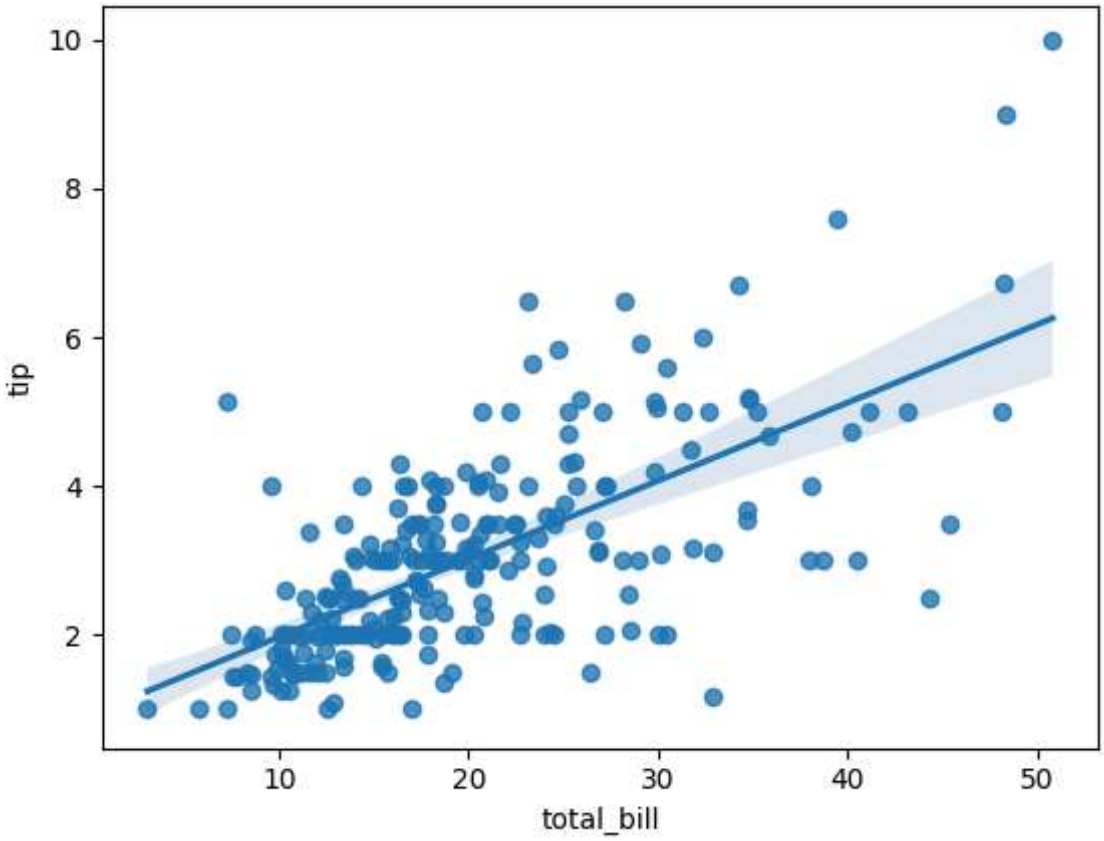
In [21]:
```
X = df[['total_bill']]
X.head()
```

Out[21]:

	total_bill
0	16.99
1	10.34
2	21.01
3	23.68
4	24.59

In [15]:
```
sns.regplot(x='total_bill', y='tip', data=df)
```

Out[15]: <Axes: xlabel='total_bill', ylabel='tip'>



prediction on the bses of single input

In [ ]:
```
X = df[['total_bill']]   # x - axis

Y = df[['tip']]   # y-axis
```

In [ ]:
```
model = LinearRegression()

model.fit(X,Y)   # model trained
```

Out[ ]:

▾ LinearRegression ⓘ ?

▸ Parameters

See the graph in svg file the prediction is exact

In [21]:
```
# lets predict the tip for an unknown total_bill

model.predict([[32]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

Out[21]:
```
array([[4.28105417]])
```

In [22]:
```
# lets predict the tip for an unknown total_bill

model.predict([[150]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

Out[22]:
```
array([[16.67394722]])
```

intermediate ML

In [5]:
```
df.head()
```

Out[5]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

multivariate linear regression

we can use multiple input features to predict the output variable. In this case, we will use both 'total_bill' and 'size' as input features to predict 'tip'.

`python`

In [6]:
```
# multivariate linear regression

x = df[['total_bill', 'size']]

y = df[['tip']]

model = LinearRegression()
```

```
model.fit(x,y)
```

Out[6]:

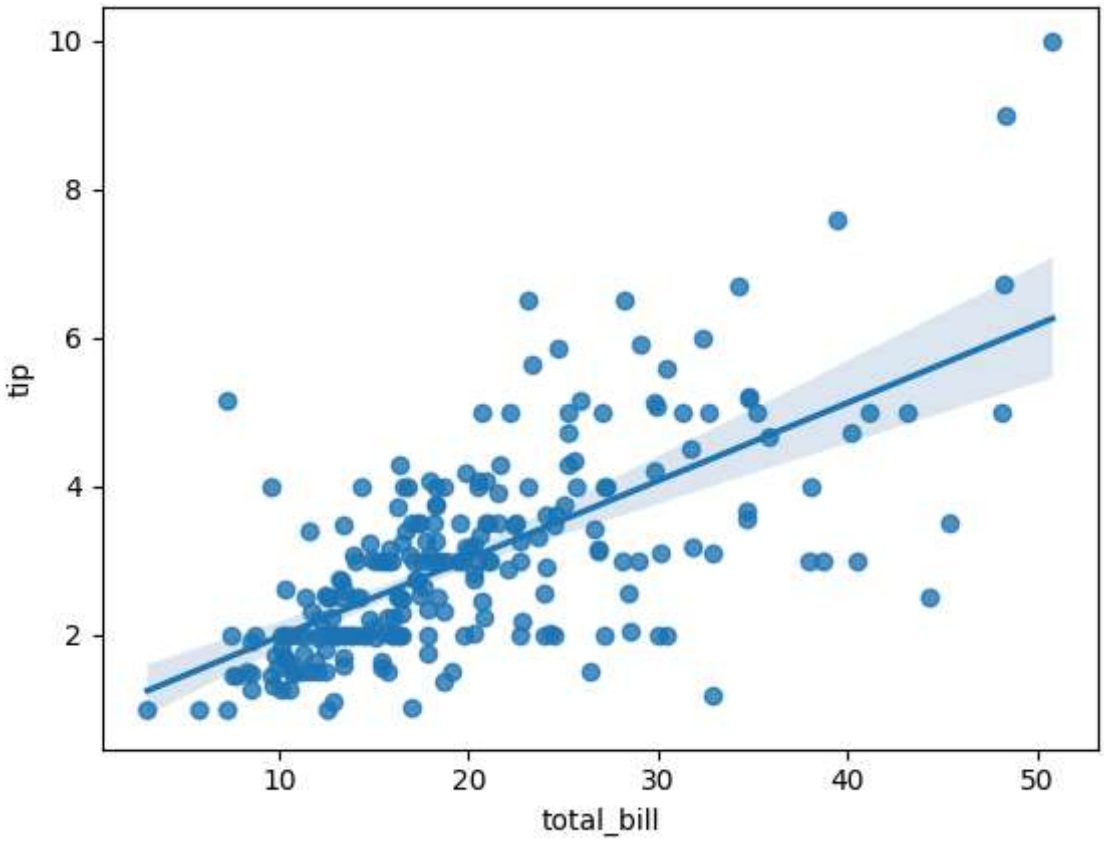
```
▾ LinearRegression
▸ Parameters
```

In [30]:

```
sns.regplot(x='total_bill', y='tip', data=df)
```

Out[30]:

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



In [7]:

```
model.predict([[32, 2]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

Out[7]:

```
array([[4.02096711]])
```

In [20]:

```
df.head()
```

Out[20]:

	total_bill	tip	sex	smoker	day	time	size	smoker_num
0	16.99	1.01	Female	No	Sun	Dinner	2	NaN
1	10.34	1.66	Male	No	Sun	Dinner	3	NaN
2	21.01	3.50	Male	No	Sun	Dinner	3	NaN
3	23.68	3.31	Male	No	Sun	Dinner	2	NaN
4	24.59	3.61	Female	No	Sun	Dinner	4	NaN

In [26]:
```
# encoding categorical variables
df = sns.load_dataset('tips')
df['smoker_num'] = df['smoker'].map({'Yes':1, 'No':0})
df.head()
```

Out[26]:

	total_bill	tip	sex	smoker	day	time	size	smoker_num
0	16.99	1.01	Female	No	Sun	Dinner	2	0
1	10.34	1.66	Male	No	Sun	Dinner	3	0
2	21.01	3.50	Male	No	Sun	Dinner	3	0
3	23.68	3.31	Male	No	Sun	Dinner	2	0
4	24.59	3.61	Female	No	Sun	Dinner	4	0

In [27]:
```
x =  df[['total_bill', 'size', 'smoker_num']]
y = df[['tip']]

model = LinearRegression()
model.fit(x,y)
```

Out[27]:
```
▾ LinearRegression
▸ Parameters
```

In [28]:
```
model.predict([[32, 2,1]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

Out[28]:
```
array([[3.99067456]])
```

classification task

In [31]:
```
df = sns.load_dataset('tips')
df.head()
```

Out[31]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [39]:
```
from sklearn.linear_model import LogisticRegression
# classification task
x = df[['total_bill', 'size', 'tip']]
```

```
# df['smoker_num'] = df['smoker'].map({'Yes':1, 'No':0})
y = df['smoker']
# model selextion
model = LogisticRegression()

# firt the model
model.fit(x,y)
```

Out[39]:

```
▾ LogisticRegression
▸ Parameters
```

In [43]:

```
model.predict([[32, 2, 1]])
model.predict([[10, 1, 0]])
model.predict([[50, 4, 1]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LogisticRegressi
on was fitted with feature names
  warnings.warn(
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LogisticRegressi
on was fitted with feature names
  warnings.warn(
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LogisticRegressi
on was fitted with feature names
  warnings.warn(
```

Out[43]:

```
array(['Yes'], dtype=object)
```

binary classification task

In [44]:

```
# binary classification
## binary classification task
df = sns.load_dataset('tips')
df.head()
from sklearn.linear_model import LogisticRegression
x = df[['total_bill', 'size', 'tip']]
y = df['smoker']
model = LogisticRegression()
model.fit(x,y)

model.predict([[32, 2,1]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but LogisticRegressi
on was fitted with feature names
  warnings.warn(
```

Out[44]:

```
array(['Yes'], dtype=object)
```

Logistic Regression

now we will use Logistic Regression for classification task. We will predict whether a person is a smoker or not based on 'total_bill', 'size', and 'tip'.

python

it uses sigmoid function

The Logistic Regression model uses the sigmoid function to map predicted values to probabilities between 0 and 1. This allows us to classify the input data into two categories (smoker or non-smoker) based on a threshold (usually 0.5).

the sigmoid function is defined as: [S(x) = \frac{1}{1 + e^{-x}}] ```python``

In [48]:
```python
# naive bayes classifier
# naive

from sklearn.naive_bayes import GaussianNB
# parameters
x = df[['total_bill', 'size', 'tip']]
# target
y = df['smoker']

# algoeirthm selection
model = GaussianNB()

# fit the model
model.fit(x,y)
# predict
model.predict([[32, 2,1]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but GaussianNB was f
itted with feature names
  warnings.warn(
```

Out[48]:
```
array(['Yes'], dtype='<U3')
```

now we will learn disicion tree classifier

In [49]:
```python
# disicion tree classifier
from sklearn.tree import DecisionTreeClassifier
x = df[['total_bill', 'size', 'tip']]
y = df['smoker']

model = DecisionTreeClassifier()

model.fit(x,y)

model.predict([[32, 2,1]])
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but DecisionTreeClas
sifier was fitted with feature names
  warnings.warn(
```

Out[49]:
```
array(['Yes'], dtype=object)
```

In [52]:
```
# disicion tree classifier
from sklearn.tree import DecisionTreeRegressor

x = df[['total_bill', 'size']]  # features

y = df['tip']   # target

model = DecisionTreeRegressor()   # model selection

model.fit(x,y)   #  fit the model


model.predict([[32, 2]])  #  predict
```

```
c:\Users\mohda\Desktop\MacLrn\.machine\Lib\site-packages\sklearn\utils\validatio
n.py:2691: UserWarning: X does not have valid feature names, but DecisionTreeRegr
essor was fitted with feature names
  warnings.warn(
```

Out[52]:
```
array([3.18])
```

and three are more algorithms for classification task like SVM, KNN, Naive Bayes etc. you can find the documentation for these algorithms on the official scikit-learn website: https://scikit-learn.org/stable/supervised_learning.html

Error Metrics for Classification

what is error metrics for classification task? the error metrics for classification task is used to evaluate the performance of a classification model. It quantifies how well the model is able to predict the correct class labels for the input data. The

there error in the model prediction can be measured using various metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. These metrics help evaluate the performance of classification models and determine how well they are able to classify the input data into the correct categories.

how can we minimize the error in classification task? we can minimize the error in classification task by using various techniques such as feature selection, hyperparameter tuning, cross-validation, and ensemble methods. Feature selection involves selecting the most relevant features that contribute to the classification task, while hyperparameter tuning involves optimizing the parameters of the model to improve its performance. Cross-validation helps to evaluate the model's performance on different subsets of the data, while ensemble methods combine multiple models to improve overall accuracy and reduce errors.

train test split

ususally split is 80% train and 20% test

we will split the data into training and testing sets to evaluate the performance of our classification model. The training set will be used to train the model, while the testing set will be used to evaluate its performance on unseen data. This helps to ensure that the model is able to generalize well to new data and is not overfitting to the training data.

root mean square error (RMSE)

The root mean square error (RMSE) is a commonly used metric to evaluate the performance of regression models. It measures the average magnitude of the errors between the predicted and actual values, with a lower RMSE indicating better model performance. RMSE is calculated by taking the square root of the average of the squared differences between the predicted and actual values.

the formulla for RMSE is: [RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}] where:

- (y_i) is the actual value
- (\hat{y}_i) is the predicted value
- (n) is the number of observations

what is confusion matrix?

A confusion matrix is a table that is used to evaluate the performance of a classification model. It summarizes the number of correct and incorrect predictions made by the model, broken down by each class. The confusion matrix typically includes four values: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These values can be used to calculate various performance metrics such as accuracy, precision, recall, and F1-score.

In [ ]:
```
#Regreession mTRetrics
import numpy as np
import pandas as pd
import seaborn as sns
```

In [55]:
```
df = sns.load_dataset('tips')
df.head()
x = df[['total_bill', 'size']]

y = df['tip']
```

In [ ]:
```
# regression matrixcs

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_

# cal lthe model
model = LinearRegression()

model.fit(x_train, y_train) # fit the model on train data
```

```
y_pred = model.predict(x_test)  # predict on test data
```

In [58]:
```
# matric to evaluate the model
from sklearn.metrics import mean_squared_error, r2_score,mean_absolute_error

mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

mae = mean_absolute_error(y_test, y_pred)
```

by the above matacies we get the idea about how well our model is performing in classification task.

evaluating the model

how can we evaluate the performance of our regression model? we can evaluate the performance of our regression model using various metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared (R2). These metrics help to quantify the difference between the predicted and actual values, and provide insights into how well the model is able to fit the data.

In [66]:
```
# we will print the metrics and see how well our model is performing
print("Mean Squared Error (MSE):", mse)
```