Monthly Intership Program for Professionals@ THE CODERS READY THE CODERS READY Data Science and Business Analytics Name: Ashima Sapehia Trainee Intern @CodersReady Task 2: Data of Students based on number of Study hours Level=Basic Problem Statement: 1. Predict the percentage of an student based on the no. of study hours. 2. This is a simple linear regression task as it involves just 2 variable. 3. As a data analyst, try to find out the prediction for score. 4. What will be the predicted score if a student studenies for 8.5hrs/day? 5. You can use Python tool to perform this analysis. In [2]: **import** pandas **as** pd import numpy as np In [1]: **import** seaborn **as** sns In [3]: import matplotlib.pyplot as plt import seaborn as sn from sklearn.linear_model import LinearRegression from sklearn.model_selection import train_test_split **%matplotlib** inline In [3]: data=pd.read_csv("student_score.txt", sep='\t') data Out[3]: **Hours Scores** 2.5 21 5.1 47 3.2 27 8.5 75 3.5 30 1.5 20 9.2 88 5.5 60 8.3 81 2.7 25 7.7 85 5.9 62 13 3.3 42 14 17 1.1 15 8.9 95 2.5 30 16 17 1.9 24 18 6.1 67 19 7.4 69 20 2.7 30 4.8 21 54 22 3.8 35 23 6.9 76 24 7.8 86 Visualization with line plot plt.style.use('fivethirtyeight') In [25]: data.plot(kind='line') plt.title('Hours vs Scores') plt.xlabel('Hours') plt.ylabel('Scores') plt.show() Hours vs Scores 80 Scores 15 Hours Data Visualization area plot In [27]: xmin=min(data.Hours) xmax=max(data.Hours) data.plot(kind='area', alpha=0.8, stacked=True, figsize=(15, 10), xlim=(xmin, xmax)) plt.title('Hours vs Scores', size=14) plt.xlabel('Hours', size=15) plt.ylabel('Scores', size=15) plt.show() Hours vs Scores Hours Scores Scores Hours Data Visualizing with scatter plot data.plot(kind='scatter', x='Hours', y='Scores', color='b', figsize=(10,8)) plt.title('Hours vs Percentage') plt.xlabel('Hours') plt.ylabel('Scores') plt.show() Hours vs Percentage 80 70 Scores 40 30 Hours By Visualization we come to know that this problem can be solved by linear regression Regression Plot sns.regplot(x= data['Hours'], y= data['Scores']) plt.title('Regression Plot', size=20 plt.ylabel('Marks Percentage', size=12) plt.xlabel('Hours Studied', size=12) plt.show() print(data.corr()) **Regression Plot** 100 80 Marks Percentage Hours Studied Hours Hours 1.000000 0.976191 Scores 0.976191 1.000000 sns.countplot(data['Hours']); C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the o nly valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(2.00 1.75 1.50 1.25 5 100 0.75 0.50 0.25 1.11.51.92.52.73.23.33.53.84.54.85.15.55.96.16.97.47.77.88.38.58.99.2sns.countplot(data['Scores']); C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the o nly valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. warnings.warn(3.0 2.5 2.0 1.0 0.5 17 20 21 24 25 27 30 35 41 42 47 54 60 62 67 69 75 76 81 85 86 88 95 Scores In [7]: import plotly.express as px import plotly.graph_objects as go In [9]: | fig=px.bar(data, x="Hours", y="Scores", title='Details') fig.show() Details 100 80 60 Scores 40 20 Hours In [11]: fig=px.histogram(data, x="Hours") fig.show() count 2 6 Hours fig=px.histogram(data, x="Scores") fig.show() 40 60 Scores fig=px.sunburst(data, path=["Hours", "Scores"]) In [14]: $\verb|C:\Pr| or amData\Anaconda3\lib\site-packages\plotly\express_core.py: 1637: Future \textit{Warning}: \\$ The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead. $\verb|C:\Pr| or amData\Anaconda3\lib\site-packages\plotly\express_core.py: 1637: Future \textit{Warning}: \\$ The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead. 35 41 4.5 4.8 54 47 60 62 In [15]: fig=px.pie(data, values='Hours', names='Scores') fig.show() 0.878% 1.2% 88 95 7.34% 30 6.94% 75 81 2.55% 86 6.78% 2.63% 85 69 3.03% 76 6.62% 67 3.59% 62 60 3.83% 47 6.23% 54 4.07% 41 35 6.15% 4.39% 42 5.91% 4.71% 27 4.87% 5.51% 25 21 Predicting the score with the single input value In [9]: hours=8.5 predicted_score=regressor.predict([[hours]]) print(f'NO. OF HOURS={hours}') print(f'predicted Score={predicted_score[0]}') NO. OF HOURS=8.5 predicted Score=[86.0031392]