****Analyzing StudentPerformance.csv****

Objective: To draw inference about whether factors like gender of the student, the race/ethnicity of the student, the level of education of their parents, the type of lunch they ate and whether the completition of test preparation course has any impact on the scores obtained by the student in the tests.

Some of the questions that this analysis will try to answer are:

1. Does the gender of student plays a role in how they perform in various courses.
2. Does the educational background of the parents impact the students performance.
3. Does the ethnicity of the student has an impact on their performance.
4. Is completing the Test Preparation course help the students in performing better.
5. Does the quality of lunch the students consume leaves an impact on how they perform.

Finally, a model will be trained to predict how the students will perform given the factors influencing their performance and will also evaluate the performance of the model.

The first step is to import all the necessary libraries needed for performing the analysis:

In [ ]:
```
#importing the necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import r2_score as r2
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
```

Exploring the 'StudentPerformance' dataset.

In [ ]:
```
#reading the StudentsPerformance.csv file and viewing it
student = pd.read_csv("/home/kasagg21/Downloads/archive (2)/StudentsPerformance.csv")
student.head()
```

As can be seen from the first five rows of the dataset, the dataset contains columns like 'gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test preparation course', 'math score', 'reading score', 'writing score'

In [ ]:
```
#Identifying the columns present in the dataset
student.columns
```

In [ ]:
```
#This displays general information about the dataset with informations like the column names their data types
#and the count of non-null values for every column.
student.info()
```

In [ ]:
```
#checking if there is any column which contains null values
student.isna().sum()
```

So as we can see that there are no null values in any column of the dataset.

Analysing the categorical variables:

In [ ]:
```
#this will help in knowing the number of categories present in each categorical variable
student.select_dtypes('object').nunique()
```

Now as we know the number of unique categories present in each of the categorical variable, it is important to see what are the unique categories present in each of them.

In [ ]:
```
#to find out the various categories present in the different categorical variable
print("Categories in 'gender' variable: ",end=" ")
print(student['gender'].unique())
print("Categories in 'race/ethnicity' variable: ",end=" ")
print(student['race/ethnicity'].unique())
print("Categories in 'parental level of education' variable: ",end=" ")
print(student['parental level of education'].unique())
print("Categories in 'lunch' variable: ",end=" ")
print(student['lunch'].unique())
print("Categories in 'test preparation course' variable: ",end=" ")
print(student['test preparation course'].unique())
```

In [ ]:
```
#This displays information about the quantitive/numerical columns, information like count, mean, standard deviation, minimum value, maximum value
#and the quartiles are displayed
student.describe()
```

We will add a new feature(column) called 'Total Score' which will be basically the sum of the scores obtained in maths, writing and reading for every student. This feature will help in better analysing the overall performance of a student.

In [ ]:
```
#Total score = math score + reading score + writing score
student['Total Score']=student['math score']+student['reading score']+student['writing score']
```

We will also add a new column 'Pass/Fail', which will basically indicate the status of the student i.e. whether they have passed(P) or failed(F). To decide whether a student have passed we are evaluatin a condition on the total score obtained by the student. We are assuming that the passing criterion if a student has a Total Score of 120 or above then they have Passed, otherwise, they Failed.

In [ ]:
```
#Criterion for getting a passing grade
def result(TS,MS,WS,RS ):
    if(TS>150 and MS>40 and WS>40 and RS>40):
        return 'P'
    else:
        return 'F'
```

In [ ]:
```
student['Pass/Fail']=student.apply(lambda x: result(x['Total Score'],x['math score'],x['writing score'],x['reading score']),axis = 1 )
```

Let's check the dataset again with the newly added two columns 'Total Score' & 'Pass/Fail'

In [ ]:
```
student.head()
```

Now using the newly added 'Pass/Fail' column, we will count the number of students passed and failed according to the passing criterion:

In [ ]:
```
#Displays the number of students passed and failed according to the passing criterion
student['Pass/Fail'].value_counts()
```

So according to the count, a total of 939 students have passed and 61 students have failed out of the 1000 students.

Now lets try to visualize the performace of the students, sometimes visualization can help in exploring underlying trends/relationships in a better way:

In [ ]:
```
plt.pie(student['Pass/Fail'].value_counts(),labels=['Pass','Fail'],autopct='%1.1f%%')
plt.title('Percentage of students Passed/Failed')
```

In [ ]:
```
sns.countplot(student['Pass/Fail'])
plt.title('Bar-plot representing the count of students passed/failed')
```

As the dataset contains both male and female students, we will try to analyze the variation of performance across the gender of the student and will try to findout if one gender performed better than the other

In [ ]:
```
# this displays the number of male and female students in the class
student['gender'].value_counts()
```

So as we can see that out of the 1000 students in the dataset, 518 are female and 482 are male. Thus the ratio of male and female students are almost uniform. Now we will try to findout how did the male & female students performed when compared to each other.

In [ ]:
```
#to find out the percentage of female students passed
print("Percentage of female students passed: {0:.2f}%"
    .format((student[(student['gender']=='female') & (student['Pass/Fail']=='P')].shape[0]/student[student['gender']=='female'].shape[0])*100))

#to find out the percentage of male students passed
print("Percentage of male students passed: {0:.2f}%"
    .format((student[(student['gender']=='male') & (student['Pass/Fail']=='P')].shape[0]/student[student['gender']=='male'].shape[0])*100))
```

Therefore from the above analysis we can observe that the male students have overall performed relatively better than the female students

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming your data is in wide-form
melted_data = student.melt(id_vars="gender", value_vars="Pass/Fail", var_name="Pass/Fail")

sns.countplot(data=melted_data, x="gender", hue="Pass/Fail")
plt.ylabel("Number of students")
plt.show()
```

We can observe from the above count plot that there is a variation between how both the genders performed and we can see that the male students have performed overall better than the female students, next we will try to analyze the performance of the students in the three different subjects and their variation across the gender.

In [ ]:
```
fig,ax = plt.subplots(3,1, figsize = (5,10))
sns.barplot(x=student['gender'],y=student['math score'], ax=ax[0], linewidth=2.5)
sns.barplot(x=student['gender'],y=student['reading score'], ax=ax[1],linewidth=2.5)
```

```
sns.barplot(x=student['gender'],y=student['writing score'], ax=ax[2],linewidth=2.5)
plt.tight_layout()
```

As can be seen from the above barplots that the male students have performed better in maths whereas the female students have relatively performed better than the male students in both reading and writing exams.

In [ ]:
```
fig,ax = plt.subplots(3,1, figsize = (5,10))
sns.boxplot(x=student['gender'],y=student['math score'],ax=ax[0])
sns.boxplot(x=student['gender'],y=student['reading score'],ax=ax[1])
sns.boxplot(x=student['gender'],y=student['writing score'],ax=ax[2])
plt.tight_layout()
```

The boxplots represent the performance of the male students vs. the performance of the female students in the three courses separately. As can be seen from the medians and the number of outliers, it can be concluded that the female students performed relatively poorer than the male students in maths but they out-performed the male students in both reading and writing scores. Thus we can conclude that, in this case the performance of a student in a course varies with the gender.

Next, trying to analyse whether the **ethnicity/race** of the student plays any role in their performance.

In [ ]:
```
#number of students belonging to each race/ethnic group
student['race/ethnicity'].value_counts()
```

Thus we can see that out of the 1000 students, 319 are from race group C, 262 are from group D,190 are from group B, 140 from group E and 89 are from the race group A. Now we will try to analyse how the students from the different race/ethnic groups have performed compared to each other.

In [ ]:
```
#number of students passed across the race/ethnic groups
print("The number of students passed across various race/ethnic group : ")
print(student['race/ethnicity'].loc[student['Pass/Fail']=='P'].value_counts())
sns.countplot(student['race/ethnicity'].loc[student['Pass/Fail']=='P'])
plt.xticks(rotation = 45)
```

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming your data is in wide-form
melted_data = student.melt(id_vars="race/ethnicity", value_vars="Pass/Fail", var_name="Pass/Fail")

sns.countplot(data=melted_data, x="race/ethnicity", hue="Pass/Fail")
plt.ylabel("Number of students")
plt.show()
```

In [ ]:
```
#to find out the percentage of students passed with the race/ethnicity  as 'group A'
print("Percentage of students passed with the race/ethnicity  as 'group A': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group A') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group A'].shape[0])*100))

#to find out the percentage of students passed with the race/ethnicity  as 'group B'
print("Percentage of students passed with the race/ethnicity  as 'group B': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group B') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group B'].shape[0])*100))

#to find out the percentage of students passed with the race/ethnicity  as 'group C'
print("Percentage of students passed with the race/ethnicity  as 'group C': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group C') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group C'].shape[0])*100))
```

```
#to find out the percentage of students passed with the race/ethnicity  as 'group D'
print("Percentage of students passed with the race/ethnicity  as 'group D': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group D') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group D'].shape[0])*100))

#to find out the percentage of students passed with the race/ethnicity  as 'group E'
print("Percentage of students passed with the race/ethnicity  as 'group E': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group E') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group E'].shape[0])*100))
```

Thus from the above analysis we can observe that the race/ethnicity group 'group E' has performed better than all other groups and the group 'group A' has performed poorer than any other groups. It can also be observed that the performance of students in race/ethinicity group gets better as we move 'group A' to 'group E'.

In [ ]:
```
fig, ax = plt.subplots(3,1, figsize=(8,12))
sns.boxplot(x=student['race/ethnicity'],y=student['math score'],ax=ax[0])
sns.boxplot(x=student['race/ethnicity'],y=student['reading score'],ax=ax[1])
sns.boxplot(x=student['race/ethnicity'],y=student['writing score'],ax=ax[2])
plt.tight_layout()
```

Thus the above observation can also be noted in the above box-plots, 'group A' has a relatively poorer performance in all the three courses whereas in comparison 'group E' performs relatively better than the other groups.

Now we will try to find the impact of the educational background of the parents on the students performance.

In [ ]:
```
#number of students having parents with various edication level
student['parental level of education'].value_counts()
```

Thus among the 1000 students, 226 students have parents with 'some college' background, 222 with 'associate's degree',196 have 'high school' background, 179 have parents with 'some high school' background, 118 with 'bachelor's degree',59 with 'master's degree' background. Now we will try to analyze how the performance of the students vary depending on their parents educational background.

In [ ]:
```
#number of students passed across the parental levels of education
print("The number of students passed across the different parental levels of education: ")
print(student['parental level of education'].loc[student['Pass/Fail']=='P'].value_counts())
sns.countplot(student['parental level of education'].loc[student['Pass/Fail']=='P'])
plt.xticks(rotation = 45)
```

In [ ]:
```
#to find out the percentage of students passed with the parental level of education as 'some college'
print("Percentage of students passed with the parental level of education as 'some college': {0:.2f}%"
    .format((student[(student['parental level of education']=='some college') & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=='some college'

#to find out the percentage of students passed with the parental level of education as 'associate's degree'
print("Percentage of students passed with the parental level of education as 'associate's degree': {0:.2f}%"
    .format((student[(student['parental level of education']=="associate's degree") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="associa

#to find out the percentage of students passed with the parental level of education as 'high school'
print("Percentage of students passed with the parental level of education as 'high school': {0:.2f}%"
    .format((student[(student['parental level of education']=="high school") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="high school"].

#to find out the percentage of students passed with the parental level of education as 'some high school'
print("Percentage of students passed with the parental level of education as 'some high school': {0:.2f}%"
    .format((student[(student['parental level of education']=="some high school") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="some high

#to find out the percentage of students passed with the parental level of education as 'bachelor's degree'
print("Percentage of students passed with the parental level of education as 'bachelor's degree': {0:.2f}%"
    .format((student[(student['parental level of education']=="bachelor's degree") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="bachelor
```

```
#to find out the percentage of students passed with the parental level of education as 'master's degree'
print("Percentage of students passed with the parental level of education as 'master's degree': {0:.2f}%"
    .format((student[(student['parental level of education']=="master's degree") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="master's d
```

In [ ]:

```
```

In [ ]:

```
plt.figure(figsize=(10,5))
plt.title("Total Score across parental level of education of students")
sns.barplot(x=student['parental level of education'],y=student['Total Score'])
```

As can be observed from the above plot that there is some influence the parent's background have on the student's performance. As can be seen, that students having parents with master's degree performed better than other and students with parents having some high school level of education performed poorer than the other groups.

Next we are going to see how the quality of lunch impacts the performance of the students:

In [ ]:

```
#number of students having 'standard' lunch vs. number of students having 'free/reduced' lunch
student['lunch'].value_counts()
```

Thus out of the 1000 students, 645 have a standard lunch and 355 have a free/reduced lunch. Now we will analyze how the type of lunch varies the performance of the students.

In [ ]:

```
#number of students passed across the type of lunch
student['lunch'].loc[student['Pass/Fail']=='P'].value_counts()
```

In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming your data is in wide-form
melted_data = pd.melt(student, id_vars="lunch", value_vars="Pass/Fail", var_name="Pass/Fail")

sns.countplot(data=melted_data, x="lunch", hue="Pass/Fail")
plt.ylabel("Number of students")
plt.show()
```

In [ ]:

```
#to find out the percentage of students passed with the lunch type as 'standard'
print("Percentage of students passed with the lunch type as 'standard': {0:.2f}%"
    .format((student[(student['lunch']=='standard') & (student['Pass/Fail']=='P')].shape[0]/student[student['lunch']=='standard'].shape[0])*100))

#to find out the percentage of students passed with the lunch type as 'free/reduced'
print("Percentage of students passed with the lunch type as 'free/reduced': {0:.2f}%"
    .format((student[(student['lunch']=="free/reduced") & (student['Pass/Fail']=='P')].shape[0]/student[student['lunch']=="free/reduced"].shape[0])*100))
```

In [ ]:

```
plt.figure(figsize=(5,5))
plt.title("Total Score across the type of lunch of the students")
sns.barplot(x=student['lunch'],y=student['Total Score'],hue=student['gender'])
```

So as we can observe from the above plot, the type of lunch has an impact on the scores of the students. The students with 'standard' lunch performed better than the student with 'free/reduced' lunch.

Now we are going to find out whether completing the 'Test Preparation Course' helps the students in performing better or not.

In [ ]:
```
#number of students who completed the 'Test preparation course' vs. the students who didn't complete the course
student['test preparation course'].value_counts()
```

Thus out of the 1000 students, 642 students didn't complete the 'Test preparation course' and 358 students completed it.

In [ ]:
```
#number of students passed across the status of completion of the test preparation course
print("The number of students passed across the status of completion of the test preparation course:")
print(student['test preparation course'].loc[student['Pass/Fail']=='P'].value_counts())
```

In [ ]:
```
#to find out the percentage of students passed with the test preparation course status as 'none'
print("Percentage of students passed with the test preparation course status as 'none': {0:.2f}%"
    .format((student[(student['test preparation course']=='none') & (student['Pass/Fail']=='P')].shape[0]/student[student['test preparation course']=='none'].shape[0])*100))

#to find out the percentage of students passed with the test preparation course status as 'completed'
print("Percentage of students passed with the test preparation course status as 'completed': {0:.2f}%"
    .format((student[(student['test preparation course']=="completed") & (student['Pass/Fail']=='P')].shape[0]/student[student['test preparation course']=="completed"].shape[0])*10
```

In [ ]:
```
plt.figure(figsize=(5,5))
sns.barplot(x=student['test preparation course'],y=student['Total Score'])
plt.title("Total Score across the status of test prep course")
plt.xlabel('Status of Test Prep Course')
```

As can be noted that the test preparation course has an impact on the performance of the students, 97.21% of the students who completed the 'Test Preparation Course'passed whereas 92.06% of the students who didn't complete 'Test Preparation Course' passed.

Now we will try to find and observe whether there is any correlation between how the students performed in the various courses.

In [ ]:
```
fig, ax = plt.subplots(3,1, figsize=(8,12))
sns.regplot(x=student['reading score'],y=student['writing score'],ax = ax[0])
sns.regplot(x=student['reading score'],y=student['math score'],ax = ax[1])
sns.regplot(x=student['writing score'],y=student['math score'],ax=ax[2])
plt.tight_layout()
```

As can be seen from the above plots that there is a strong correlation between the scores. To visualize the correlation in a better way, we produce a heat-map:

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select numeric columns from your DataFrame
numeric_columns = student.select_dtypes(include='number')

# Calculate the correlation matrix
correlation_matrix = numeric_columns.corr()

# Create a heatmap
sns.heatmap(correlation_matrix, cmap="Reds", annot=True)
plt.xticks(rotation=90)
plt.show()
```

As can be observed from the above heat-map that there is a strong correlation between 'reading score' and 'writing score'. The 'math score' is also correlated with the 'reading score' and 'writing score'

So as we have analysed the impact of different features on the student's performance and we observed that factors like 'gender', 'race/ethinicty', 'lunch', 'test preparation course' and 'parental level of education' impacted the scores obtained by the students.

Now we will try to train a model to be able to predict the **'Pass/Fail'** status of students provided with the features impacting the score of the student.

In [ ]:
```
student.head()
```

In [ ]:
```
X=student[['gender','race/ethnicity','parental level of education','lunch','test preparation course']]
X.head()
```

As we know to train a model with categorical variables, they must be first converted into a form which can be utilized for the model fitting purpose. We have used the One Hot Encoding technique to transform the categorical variables.

In [ ]:
```
X_category = student[['gender','race/ethnicity','parental level of education','lunch','test preparation course']]
```

In [ ]:
```
# Applying one-hot encoding to each column with categorical data
OH_encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
```

In [ ]:
```
X_OH = pd.DataFrame(OH_encoder.fit_transform(X_category))
X_OH.index = X.index #One-hot encoding removes the index so it's necessary to put them back
X_OH.head()
```

In [ ]:
```
#collecting the total score of the students from the dataset
y=student['Pass/Fail']
y.head()
```

In [ ]:
```
lb=LabelEncoder()
y=lb.fit_transform(y)
```

In [ ]:
```
# Divide data into training and validation subsets
X_train, X_valid, y_train, y_valid = train_test_split(X_OH, y, train_size=0.8, test_size=0.2,random_state=0)
```

In [ ]:
```
model = RandomForestRegressor()
model.fit(X_train,y_train)
```

In [ ]:
```
#model predicting
preds=model.predict(X_valid)#predictions made by the model
```

In [ ]:
```
preds= np.where(preds<0.4,0,1)
```

In [ ]:
```
preds
```

In [ ]:
```
y_valid
```

Now we will try to evaluate the performace of our model by calculating the Mean Absolute Error value and the R2 value

In [ ]:
```
#Calculating the Mean Absolute Error value
mae(y_valid,preds)
```

For better evaluation, we will perform cross validation and will try to obtain the Mean Absolute Error(MAE) value:

In [ ]:
```
scores = -1 * cross_val_score(model, X_OH, y,cv=5,scoring='neg_mean_absolute_error')
print("MAE scores:\n", scores)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix

# creating a confusion matrix
cm = confusion_matrix(y_valid, preds)

# printing the confusion matrix
plt.rcParams['figure.figsize'] = (8, 8)
sns.heatmap(cm, annot = True, cmap = 'Reds')
plt.show()


# Calculate precision, recall, and F1 score
precision = precision_score(y_valid, preds)
recall = recall_score(y_valid, preds)
f1 = f1_score(y_valid, preds)

print(f"Confusion Matrix:\n{cm}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

In [ ]:
```
from sklearn.metrics import accuracy_score

# Calculate the accuracy of the model
accuracy = accuracy_score(y_valid,preds)

print(f"Accuracy: {accuracy:.2%}")
```

In [ ]:
```
custom_data = {
    'gender': 'female',
    'race/ethnicity': 'group B',
    'parental level of education': 'some collage',
    'lunch': 'free/reduced',
    'test preparation course': 'none'
}
```

In [ ]:
```
# Convert custom input data dictionary to a Pandas DataFrame
custom_data_df = pd.DataFrame([custom_data])

# Apply one-hot encoding to the categorical variables in the custom data
custom_data_encoded = pd.DataFrame(OH_encoder.transform(custom_data_df))
custom_data_encoded.index = custom_data_df.index

# Now, custom_data_encoded contains the preprocessed data that can be used for prediction.
```

In [ ]:
```
# Use the trained model to make predictions on the preprocessed custom input data
predictions = model.predict(custom_data_encoded)

# The 'predictions' variable now contains the model's prediction.
```

In [ ]:
```
# Determine pass/fail based on a threshold value
pass_fail_prediction = 1 if predictions[0] >= 0.66 else 0

# The 'pass_fail_prediction' variable now contains the pass/fail prediction.
```

In [ ]:
```
# Assuming you have determined pass_fail_prediction as mentioned in the previous response
print("Pass/Fail Prediction:", pass_fail_prediction)
```

In [ ]:
```
import joblib

# Your trained model (already defined in your code)
model = RandomForestRegressor()

# Save the model to a file
joblib.dump(model, '/home/kasagg21/Downloads/archive (2)/trained_model_v3.joblib')
```

In [ ]:
```

```