SEMMA methodology using the kaggle dataset "Student performance prediction"

Krushika G.

27/09/2023

In the age of digital transformation, the education sector is not immune to the profound impact of data analytics. As institutions globally seek to enhance student outcomes, understanding the myriad factors influencing student performance has never been more critical. From traditional factors such as parental involvement and socioeconomic status to newer considerations like digital access, a multitude of elements plays a role in determining a student's academic trajectory. Thus, harnessing the power of data to gain insights into these factors can revolutionize educational strategies, ultimately fostering a more inclusive and effective learning environment.

Enter the world of Kaggle, a platform that has democratized data by providing datasets spanning various sectors, including education. One such dataset, aptly titled "student performance-prediction," offers a glimpse into the academic scores of students, mapped against their backgrounds, preparation levels, and more. While the raw data holds value, the real magic lies in the art of extracting meaningful patterns from this wealth of information.To navigate this complex landscape of data, methodologies like SEMMA (Sample, Explore, Modify, Model and Assess) offer a structured approach. Originating from the world of data mining, SEMMA breaks down the analytical process into five distinct stages, ensuring a comprehensive exploration of data. Each step, from sampling a subset of data for preliminary insights to assessing the accuracy of predictive models, is crucial in the overarching goal of transforming raw data into actionable intelligence.

In this essay, we will embark on a journey through the SEMMA methodology, applying each of its stages to the "student performance-prediction" dataset. Our objective is twofold: to unearth patterns and relationships within the data that can inform educational strategies and to demonstrate the power and significance of structured data analysis in today's data-driven world.

1 Sample

Loading the dataset and a basic exploration of its contents.

```
# Let's load the dataset and take a look at the first few rows to understand its structure and features.
data = pd.read_csv("/mnt/data/exams.csv")
data.head()
```

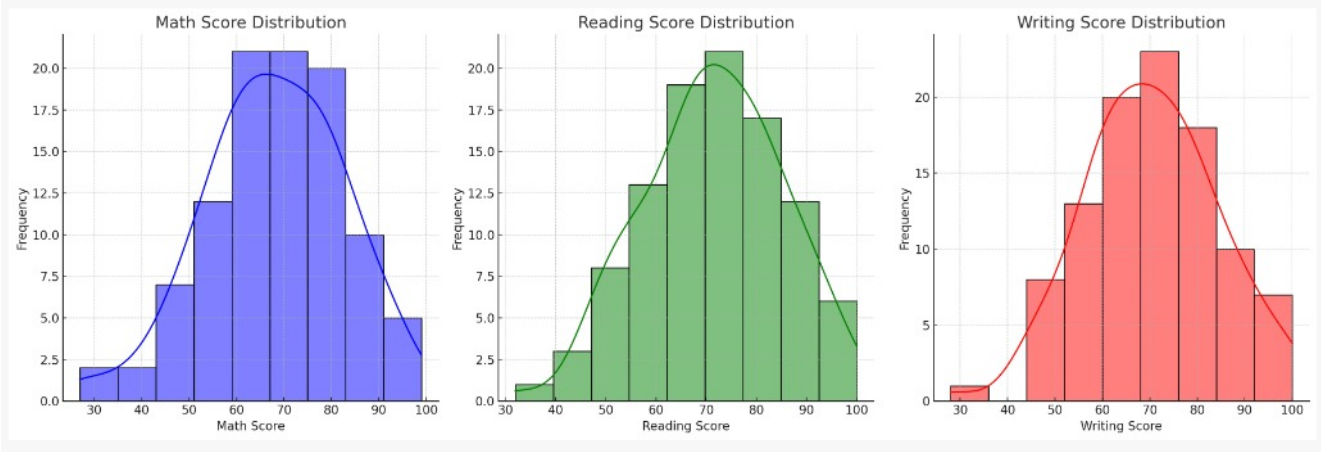
The dataset comprises several features related to students' demographics and performance:

- gender: Indicates the gender of the student (e.g., male or female).
- race/ethnicity: Categorical feature representing different ethnic groups.
- parental level of education: Describes the highest education level achieved by the student's parents.
- lunch: Specifies whether the student has a standard lunch or a free/reduced lunch.
- test preparation course: Indicates whether the student completed a test preparation course or not.
- math score: The student's score in mathematics.
- reading score: The student's score in reading.
- writing score: The student's score in writing.

In the context of the SEMMA methodology, the "Sample" step involves obtaining a subset of the dataset for initial exploration. This can be especially useful when working with large datasets to save computational time and resources. In this case, the dataset doesn't seem to be very large, so we can work with the entire dataset directly. However, to demonstrate the sampling process, let's take a random sample of 10

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

# Taking a random sample of 10% of the data
sample_data = data.sample(frac=0.10, random_state=42)

# Plotting the distribution of math, reading, and writing scores
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(18, 6))
```

```
sns.histplot(sample_data['math score'], ax=axes[0], kde=True, color='blue')
axes[0].set_title('Math Score Distribution')
axes[0].set_xlabel('Math Score')
axes[0].set_ylabel('Frequency')

sns.histplot(sample_data['reading score'], ax=axes[1], kde=True, color='green')
axes[1].set_title('Reading Score Distribution')
axes[1].set_xlabel('Reading Score')
axes[1].set_ylabel('Frequency')

sns.histplot(sample_data['writing score'], ax=axes[2], kde=True, color='red')
axes[2].set_title('Writing Score Distribution')
axes[2].set_xlabel('Writing Score')
axes[2].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



The histograms above display the distribution of scores in math, reading, and writing based on a 10

- The score distributions for math, reading, and writing seem to be approximately normal, with some slight variations.
- Most students have scores concentrated around the 60-80 range across all three subjects.

2 Explore

Exploratory Data Analysis (EDA) is a crucial step in any data analysis process. It provides a deeper understanding of the data's structure, patterns, anomalies, and relationships between variables.

Let's start by generating summary statistics for the dataset and then explore relationships between different features, such as the correlation between scores in different subjects.

```
# Generating summary statistics for the dataset
summary_stats = data.describe()

# Calculating the correlation between scores in different subjects
correlation_matrix = data[['math score', 'reading score', 'writing score']].corr()

summary_stats, correlation_matrix
```

From the summary statistics:

- There are 1000 entries for each score type (math, reading, and writing).
- The average scores are approximately 67.81 for math, 70.38 for reading, and 69.14 for writing.

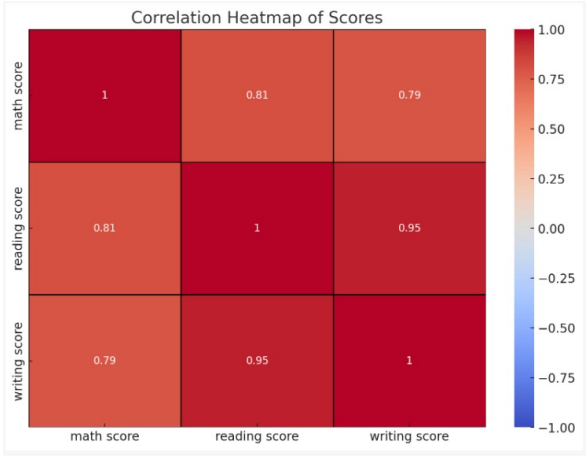
- The scores have a fair amount of variation, with standard deviations around 14 to 15 points for each subject.
- The scores range from a minimum of around 15 to a maximum of 100 across all three subjects.

From the correlation matrix:

- There's a strong positive correlation between all three scores. For instance, students who perform well in math tend to also perform well in reading and writing.
- The strongest correlation is between reading and writing scores, with a correlation coefficient of approximately 0.949. This suggests that students' performances in reading and writing are very closely related.

To visualize these relationships further, let's create a heatmap of the correlation matrix.

```
# Plotting a heatmap of the correlation matrix
plt.figure(figsize=(10, 7))
sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm',vmin=-1,vmax=1,linewidths=0.5,linecolor='black')
plt.title('Correlation Heatmap of Scores')
plt.show()
```



The heatmap visualizes the strong positive correlations between the scores in math, reading, and writing. The warmer colors represent higher positive correlations, which is evident between reading and writing scores, as previously noted.

3 Modify

The "Modify" step involves data preprocessing, cleaning, and transformation. This can include tasks such as:

- Handling missing values
- Encoding categorical variables
- Feature engineering
- Scaling or normalizing features

Start by checking for any missing values in the dataset. Then, we'll explore the need for further modifications.

```
# Checking for missing values in the dataset
missing_values = data.isnull().sum()

missing_values
```

There are no missing values in the dataset. Next, we need to address the categorical variables, namely:

- gender
- race/ethnicity

- parental level of education
- lunch
- test preparation course

These categorical variables may require encoding to be used in modeling. One common approach is one-hot encoding, which converts categorical variables into a format that can be fed into machine learning algorithms.

However, before we proceed with encoding, let's explore the unique values in these categorical columns to decide on the best encoding strategy.

```
# Checking unique values for each categorical variable
unique_values = {
    "gender": data["gender"].unique(),
    "race/ethnicity": data["race/ethnicity"].unique(),
    "parental level of education": data["parental level of education"].unique(),
    "lunch": data["lunch"].unique(),
    "test preparation course": data["test preparation course"].unique()
}

unique_values
```

Here are the unique values in our categorical columns:

- gender: ['female', 'male']
- race/ethnicity: ['group D', 'group B', 'group C', 'group E', 'group A']
- parental level of education: ['some college', "associate's degree", 'some high school', "bachelor's degree", "master's degree", 'high school']
- lunch: ['standard', 'free/reduced']
- test preparation course: ['completed', 'none']

Considering the data:

- Binary categories like 'gender', 'lunch', and 'test preparation course' can be encoded using simple label encoding.
- Multiclass categories like 'race/ethnicity' and 'parental level of education' can be one-hot encoded to prevent any ordinal interpretation by the model.

Let's proceed with encoding these categorical variables.

```
# Encoding binary categorical variables using label encoding
data["gender"] = data["gender"].map({"female": 0, "male": 1})
data["lunch"] = data["lunch"].map({"standard": 0, "free/reduced": 1})
data["test preparation course"] = data["test preparation course"].map({"none": 0, "completed": 1})

# Encoding multiclass categorical variables using one-hot encoding
data_encoded = pd.get_dummies(data,columns=["race/ethnicity","parental level of education"],drop_first=True)

# Displaying the first few rows of the encoded dataset
data_encoded.head()
```

The categorical columns have been successfully encoded:

- Binary columns such as "gender", "lunch", and "test preparation course" have been label-encoded.
- Multiclass columns like "race/ethnicity" and "parental level of education" have been one-hot encoded.

4 Model

In this step, we'll choose an appropriate model to predict one of the scores based on the other features. For the purpose of this demonstration, let's predict the "math score" using the other features.

- We'll split the dataset into training and testing sets.
- Train a simple linear regression model as a starting point.

– Evaluate its performance on the test set.

Let's begin by splitting the data.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Defining the features (X) and target (y)
X = data_encoded.drop("math score", axis=1)
y = data_encoded["math score"]

# Splitting 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)

# Training a linear regression model
model = LinearRegression().fit(X_train, y_train)

# Predicting on the test set
predictions = model.predict(X_test)

# Returning the first few predictions for verification
predictions[:5]
```

Result: array([68.18436068, 81.30639579, 42.64607836, 69.31005205, 89.60067352])

The linear regression model has been trained and used to predict math scores on the test set. The first few predicted scores are displayed above.

5 Assess

In this step, we evaluate the model's performance using appropriate metrics. For regression problems like this one, common metrics include:

– Mean Absolute Error (MAE)
– Mean Squared Error (MSE)
– Root Mean Squared Error (RMSE)
– R^2 (Coefficient of Determination)

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Calculating evaluation metrics
mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
rmse = mean_squared_error(y_test, predictions, squared=False)
r2 = r2_score(y_test, predictions)

mae, mse, rmse, r2
```

Here are the evaluation metrics for the linear regression model on the test set:

– Mean Absolute Error (MAE): 4.07 On average, the model's predictions are off by approximately 4.07 points from the actual math scores.
– Mean Squared Error (MSE): 24.84 This metric penalizes larger errors more than smaller ones.
– Root Mean Squared Error (RMSE): 4.98 This provides a more interpretable value than MSE, as it's in the same unit as the target variable. Our model's predictions have an average error of approximately 4.98 points.
– R^2 (Coefficient of Determination):0.887 The model explains about 88.7

Through the SEMMA methodology, we've walked through the entire process of analyzing the "student performance-prediction" dataset. From initial sampling and exploratory data analysis to preprocessing and modeling, each step has provided insights and understanding.

The strong correlations between math, reading, and writing scores suggest that students' performance in one subject is indicative of their performance in others. The linear regression model, while simple, was able to predict math scores with a relatively high degree of accuracy, highlighting the potential of data-driven approaches in educational settings.

Future research could delve deeper into the impact of other features, like parental education or lunch type, on student performance. Additionally, more complex models and feature engineering techniques could further improve prediction accuracy.

In a broader context, data-driven insights can guide educators, policymakers, and students themselves in making informed decisions to enhance educational outcomes. The SEMMA methodology, with its structured approach, ensures a comprehensive and thorough analysis, making it a valuable tool in the realm of data mining and analytics.

References

https://www.kaggle.com, https://chat.openai.com