TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Scope of Topic

Data science with AIML (Artificial Intelligence & Machine Learning) isn't a single, well-defined topic, but rather a powerful intersection that widens the scope of both fields. Data science provides the tools and techniques to extract knowledge from data, while AIML offers advanced algorithms for computers to learn and make intelligent decisions. Together, they unlock a vast potential. This encompasses tasks like uncovering hidden patterns in complex datasets, building predictive models for real-world applications, and even enabling machines to understand and respond to natural language. The scope of data science with AIML continues to grow as it tackles challenges in diverse areas like healthcare, finance, and manufacturing, ultimately shaping a future driven by data-powered insights and intelligent automation.

1.2 Importance of Internship

An internship in data science with AIML (Artificial Intelligence & Machine Learning) is a springboard to a flourishing career. It equips you with the practical skills and real-world experience that employers crave. You'll gain hands-on experience working with data, building and evaluating models, and translating complex concepts into practical solutions. This not only bolsters your technical proficiency but also allows you to discover your niche within the vast field. Furthermore, internships provide invaluable networking opportunities, connecting you with industry professionals who can mentor you and guide your future career path. In a field driven by innovation, an internship in data science with AIML is your chance to get ahead of the curve and lay the groundwork for a successful and impactful future.

1.3 Relation to Previous Work and Present Development

The internship at Cranes Varsity built upon the foundational knowledge I acquired during my academic studies. For example, I had previously learned the basics of programming languages, data sets in my coursework. However, the internship provided a deeper understanding and practical application of these concepts. Developing the student performance project was an excellent opportunity to integrate these skills into a cohesive.

Chapter 2

BACKGROUND OF CRANES VARSITY

2.1 Company Overview

Cranes Varsity is a pioneer in technical training and educational technology services, with a legacy spanning over 25 years. Initially established as a technical training institute, Cranes Varsity has evolved into a comprehensive EdTech platform dedicated to bridging the gap between academia and industry. A division of Cranes Software International Ltd, Cranes Varsity focuses on empowering aspiring professionals through rigorous and high-impact training programs. The institute offers a wide array of technology education services tailored to the needs of graduates, universities, working professionals, corporate entities, and defence sectors.

2.2 Market Position

Cranes Varsity holds a prominent position in the EdTech and technical training market. It has established itself as a trusted partner for over 5,000 reputed academic institutions, corporate organizations, and defence entities. Through its extensive network of over 2,000 universities and colleges, Cranes Varsity has successfully trained more than 100,000 engineers and placed over 70,000 in major Indian and multinational corporations. The institute's alumni network exceeds 50,000, with many serving as ambassadors of the Cranes Varsity brand worldwide. This extensive reach and proven track record.

2.3 Contributions to the EdTech Industry

Cranes Varsity has made significant contributions to the EdTech industry through its innovative training programs and commitment to student success. As an authorized training partner for industry leaders such as Texas Instruments, MathWorks, Wind River, and ARM, Cranes Varsity provides specialized training in areas like embedded systems, MATLAB, digital signal processing (DSP), and more. The institute has expanded its training domains to include emerging industry trends such as automotive systems, the Internet of Things (IoT), VLSI design, data science with aiml with Java, data science, and business analytics.

Cranes Varsity operates under the principle "We Assist Until We Place," demonstrating a steadfast dedication to participant satisfaction and placement. The institute's competitive advantage lies in its array of high-end technology training programs and its unique "Hire, Train & Deploy" model, which ensures that corporate partners receive well-trained, job-ready professionals. Cranes Varsity's educational approach, known as EEE (Educate, Evolve, Employment), integrates advanced pedagogical practices with Learning Management Systems (LMS) to deliver focused, hands-on training by subject-matter experts and professionals.

Chapter 3

Internship Process

3.1 Training in Data science and AIML

My internship at Cranes Varsity was an immersive journey into the world of full stack development. The training was well-structured, covering both front-end and back-end development, which gave me a thorough understanding of how to build complete web applications from scratch.

3.2 Python

The company gave us the Python programming course with a captivating introduction. They likely started by explaining the history and versatility of Python, highlighting its readability and ease of use compared to other languages. They might have then delved into Python's core strengths, like its extensive libraries for data science, web development, and automation. The introduction would have undoubtedly sparked your interest by showcasing the vast applications of Python and how it can empower you to achieve your programming goals.

Python reigns supreme as the language of choice for data science and AIML (Artificial Intelligence & Machine Learning) for several compelling reasons:Python's syntax is known for its clarity and resemblance to natural language. This makes it easier to learn and write code, even for beginners with no prior programming experience. This is a huge advantage, especially in data science and AIML, where complex concepts need to be translated into code efficiently.The wealth of Python libraries specifically designed for data science and AIML tasks is unmatched. Libraries like NumPy for numerical computing, Pandas for data manipulation and analysis, Matplotlib and Seaborn for data visualization, and Scikit-learn for machine learning algorithms form a powerful toolkit for data scientists. These libraries provide pre-built functions and tools, saving you time and effort in reinventing the wheel. Python's flexibility allows it to handle various aspects of a data science or AIML project. You can use it for data cleaning, wrangling, analysis, model building, deployment, and even creating interactive visualizations. This eliminates the need to switch between multiple languages for different stages of the project.

3.3 Skills Acquired

Over the course of the internship, I gained a host of technical and professional skills that are vital for a career in DATA SCIENCE WITH AIML.

3.3.1 Technical Skills

An effective data science with AIML course will equip you with a valuable blend of skills that bridge the gap between data analysis and artificial intelligence. Here's a breakdown of the key areas we likely developed:

- **Data Wrangling and Cleaning:** Learn to acquire data from various sources, handle missing values, and manipulate data into a usable format for analysis. This involves tools like Pandas and SQL.
- **Statistical Analysis:** Gain a solid understanding of statistical concepts like hypothesis testing, correlation, and regression analysis. This foundation helps you interpret data insights and draw meaningful conclusions.
- **Data Visualization:** Develop the ability to create informative and visually appealing charts and graphs to effectively communicate data stories. Libraries like Matplotlib and Seaborn are commonly used.

3.3.2 Professional Skills

- **Project Management**: I learned how to plan, execute, and manage a data science with aiml project from start to finish.
- **Problem-Solving**: I improved my ability to troubleshoot and solve problems efficiently during the development process.
- **Collaboration**: Working with mentors and peers helped me enhance my teamwork and communication skills.
- **Time Management**: I developed the ability to prioritize tasks and manage my time effectively to meet project deadlines.

3.4 Experiences Gained

The practical experiences I gained during the internship were incredibly valuable and really brought the theoretical knowledge to life.

3.4.1 Project Work

One of the highlights was working on a real-world project: developing a data science with aiml. This project was a fantastic opportunity to apply everything I had learned.

Planning and Design:We started by gathering requirements and planning the student analysis project. Designing the user interface was a creative process that set the stage for development.

Development: The development process for a student analysis data science with AIML project can be broken down into several key stages:

- **Identify the specific area of student analysis you want to focus on:** This could be predicting academic performance, identifying factors affecting student engagement, or recommending personalized learning resources.
- **Set clear and measurable goals for your project:** What insights do you want to gain? How will your project benefit students or educators?

- **Gather relevant student data:** This might involve working with educational institutions to access anonymized student records, including grades, attendance data, demographic information, and potentially survey responses.
- **Explore and understand the data:** Use data cleaning techniques to address missing values and inconsistencies. Analyze the data to identify patterns and trends. Tools like Pandas and exploratory data analysis (EDA) techniques are helpful here.
- **Create new features from existing data:** This could involve calculating ratios, creating categorical variables, or transforming text data for analysis. This step is crucial for building robust machine learning models.
- **Choose appropriate AIML techniques:** Based on your goals, you might consider supervised learning algorithms like decision trees or regression models for prediction tasks, or unsupervised learning techniques like clustering to identify student groups.
- **Train and evaluate the model:** Split your data into training and testing sets. Train your model on the training data and evaluate its performance on the unseen testing data. Tools like Scikit-learn are commonly used for machine learning in Python.
- **Interpret the results:** Analyze how well your model performs and what insights it reveals about student performance or related factors.

- **Visualize your findings:** Create clear and informative charts and graphs to effectively communicate your insights to a broader audience.
- **Develop a prototype or application:** If your project involves building a system to recommend resources or predict performance, consider developing a user-friendly interface for deployment.
- **Measure the impact of your project:** Evaluate how your findings and developed system are improving student outcomes or decision-making processes.

Testing and deployment:

- **Data Quality Checks:** Ensure data accuracy and completeness by checking for missing values, outliers, and inconsistencies. Tools like Pandas profiling can be helpful.
- **Data Leakage Prevention:** Verify your model isn't "cheating" by using information not available during real-world predictions. Split your data into training, validation, and testing sets to avoid this.
- **Performance Evaluation:** Use appropriate metrics like accuracy, precision, recall, F1-score, or RMSE (depending on your project goals) to assess your model's effectiveness on the testing dataset.
- **Cross-Validation:** Train your model on different subsets of your data to ensure its generalizability and avoid overfitting.

3.5 Mentorship and Feedback

Throughout the internship, I received continuous support and feedback from experienced mentors at Cranes Varsity. They guided me through challenges and helped me refine my skills. Regular review sessions provided invaluable insights into best practices in full stack development, and their constructive feedback was crucial for my growth.

Chapter 4

Project: Student performance analysis

4.1 Project Description

As part of my internship at cranes varasity, I developed a student performance analysis system. This project aims to utilize data science and Artificial Intelligence & Machine Learning (AIML) techniques to analyze student performance data and gain valuable insights to improve academic outcomes. By analyzing historical data and identifying key factors that influence student success

4.2 Technologies Used

Several key technologies come together to build a student performance analysis system using data science and AIML (Artificial Intelligence & Machine Learning):

Programming Languages:

- **Python:** The reigning champion for data science and AIML. Its readability, extensive libraries, and versatility make it ideal for data manipulation, analysis, model building, and visualization.

Libraries:

- **Pandas:** For data cleaning, manipulation, and exploration.
- **NumPy:** For numerical computations and array operations.
- **Scikit-learn:** Provides a comprehensive toolkit for machine learning algorithms like decision trees, random forests, and support vector machines.
- **Matplotlib/Seaborn:** For creating informative and visually appealing data visualizations.

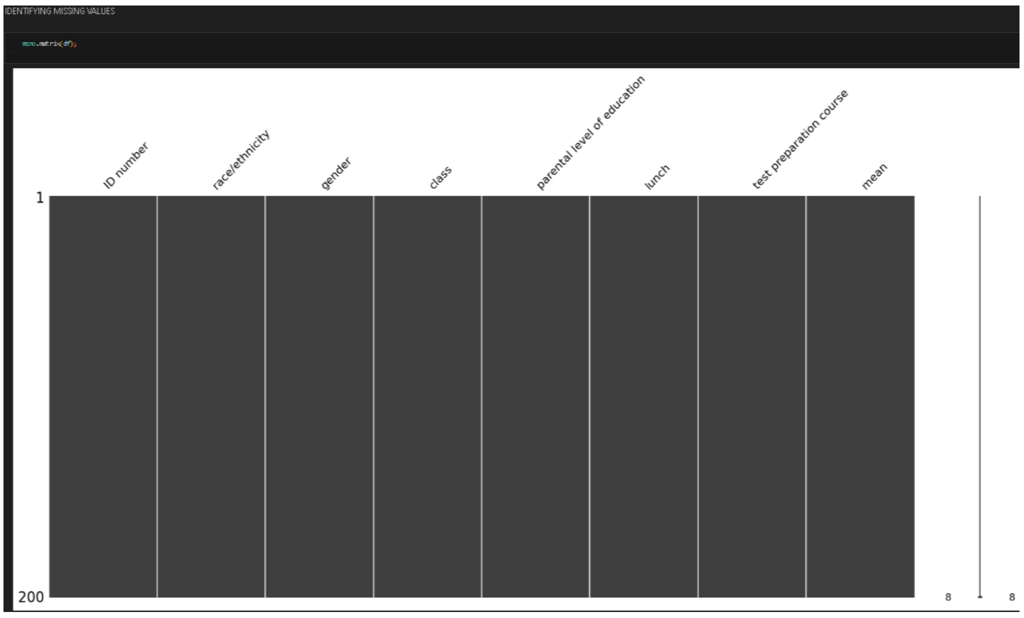
4.3 Development Process

The development process was an exciting journey that involved several key stages:

During my internship at cranes varacity, I developed an student performance analysis using the python ,data science and aiml. The project aimed to provide the visualization and the performance analysis when there are various factors.

1. Import Libraries:

- **pandas (pd):** As mentioned earlier, pandas is the workhorse for data analysis in Python. It provides data structures like DataFrames (similar to spreadsheets) and Series (single-dimensional arrays) that efficiently handle student information.
- **missingno (msno):** (assuming it's installed) This library offers functionalities to identify and visualize missing data within the DataFrame. This is crucial because missing values can significantly impact analysis.
- **numpy (np):** NumPy provides powerful tools for numerical computations. While not directly used here for initial exploration, it might be needed for calculations or data transformations later.
- **matplotlib.pyplot (plt) and seaborn (sns):** These libraries handle data visualization. Matplotlib offers a flexible plotting framework, and Seaborn builds upon it to create aesthetically pleasing and informative statistical plots. The %matplotlib inline magic command (within Jupyter Notebook) ensures plots are displayed directly within the notebook.


IDENTIFYING MISSING VALUES

```
msno.matrix(df);
```

Figure 4.1 Identify missing value

2. Read CSV Data:

- df = pd.read_csv("/content/Student data.csv") reads the "Student data.csv" file, located on your Google Colab storage (assuming it's uploaded there), into a pandas DataFrame named df. This DataFrame holds the student information in a structured format, making analysis easier.

3. Data Shape and Content:

- **df.shape:** This displays the dimensions (number of rows and columns) of the DataFrame df. It helps understand the size and structure of your data. For instance, it might return (100, 5), indicating there are 100 student records (rows) and 5 data points (columns) for each student.
- **df:** Printing df by itself usually shows the first few rows of the DataFrame. This provides a quick glimpse at the student data and its column names. You can adjust the number of rows displayed using pd.set_option('display.max_rows', n), where n is the desired number.

4. Data Exploration Techniques:

Now that w have the data loaded, here are some ways that we can explore it further:

a) Check for Missing Data:

- **msno.matrix(df):** This generates a heatmap visualizing missing values across the DataFrame. Darker cells represent more missing data in that specific feature. Use this to identify features with significant missing values that might require cleaning or handling strategies.

b) Get Descriptive Statistics:

- **df.describe():** This provides summary statistics like mean, standard deviation, minimum, maximum, percentiles etc. for numerical columns (e.g., marks, age). This helps understand the central tendency, spread, and potential outliers in the data.

c) Explore Specific Features:

- Access individual columns using their names within square brackets. For example, df["math"] retrieves the math scores of all students. You can further analyze specific columns using descriptive statistics or visualizations.

d) Visualize Data:

- **Histograms:** Use plt.hist(df["math"]) or sns.histplot(df["math"]) to visualize the distribution of math scores. This helps identify skewness, potential outliers, and **understand** the spread of data.
- **Scatter Plots:** Create scatter plots using plt.scatter(df["math"], df["science"]) or sns.scatterplot(x="math", y="science", data=df) to explore the relationship between two features (e.g., math vs science scores). This might reveal correlations or patterns.
- **Box Plots:** Use plt.boxplot(df[["math", "science"]]) or sns.boxplot(x = "subject", y = "score", data=df.melt()) to compare the distribution of scores across different subjects (assuming there's a subject column). Box plots visually represent the median, interquartile range (IQR), and potential outliers in each subject.

5. Warnings:

- **import warnings:** This line imports the warnings module, which provides functionalities for handling warnings that might arise during program execution.
- **warnings.filterwarnings("ignore"):** This line sets the filtering mode for warnings using the filterwarnings function of the warnings module. Here, "ignore" is passed as an argument, instructing Python to ignore all warnings from that point onwards.

Use of this Code to supress the warning:

- **Temporary Suppression During Development:** Sometimes, warnings can be disruptive during development, especially when you're actively modifying code and might be triggering warnings due to temporary inconsistencies.

6. df.describe() :

Method in pandas is a powerful tool for getting a quick summary of the numerical columns in your DataFrame. Here's a breakdown of what it does and how it helps you understand your data:

Functionality: When you call df.describe(), it calculates various statistics for each numerical column (numeric data types like integers or floats) in your DataFrame df. These statistics provide insights into the central tendency, spread, and distribution of the data within those columns.

Output:df.describe() typically returns a new DataFrame with the following information for each numerical column:

- **count:** The number of non-null values in the column. This indicates how many data points have actual values (not missing).
- **mean:** The average value in the column.
- **std:** The standard deviation, which represents how spread out the data is from the mean.
- **min:** The minimum value in the column.
- **25%:** The first quartile (Q1), which is the value below which 25% of the data lies.
- **50%:** The median (Q2), which is the middle value when the data is sorted.
- **75%:** The third quartile (Q3), which is the value above which 75% of the data lies.
- **max:** The maximum value in the column.
- **Central Tendency:** The mean and median indicate the "center" of the data distribution. Are they similar, suggesting a symmetrical distribution? Or is there a skew towards higher or lower values?
- **Spread:** The standard deviation tells you how much the data deviates from the mean. A larger standard deviation indicates a wider spread of data points.
- **Outliers:** The minimum and maximum values can reveal potential outliers, data points significantly different from the rest.

7. **df.info():**

The df.info() method in pandas is a concise way to get an informative summary of your DataFrame, df in this case. Here's a breakdown of what it typically displays and how it helps you understand your data:

- **Data Type:** It shows the data type (e.g., integer, float, object) of each column in your DataFrame. This helps you understand how the data is stored and the kind of operations you can perform on it.
- **Non-Null Count:** This indicates the number of non-null (non-missing) values in each column. It's crucial to identify potential missing data that might require cleaning or handling strategies.
- **Memory Usage:** It displays an estimate of the memory footprint of your DataFrame. This can be helpful, especially when working with large datasets, to gauge memory consumption.
- **Index:** It specifies the type of indexing used for the rows in your DataFrame. Common indexing methods include RangeIndex (default, used in your case) and others like custom indexing schemes.

8. msno.matrix Function:

The msno.matrix function generates a heatmap visualization that represents missing values in your DataFrame. Here's what the heatmap typically depicts:

- **Colors:** Darker colors generally indicate a higher number of missing values in that specific cell. Lighter colors represent fewer missing values or potentially no missing data.
- **Columns:** Each column in the heatmap corresponds to a feature (column) in your DataFrame.
- **Rows:** By default, rows represent observations (rows) in your DataFrame. However, msno.matrix can also be configured to display missing value patterns across categories within categorical features.
- **Visual Identification:** The heatmap provides a quick and clear visual representation of missing data patterns, making it easier to identify features with significant missing values compared to inspecting raw data.
- **Decision-Making:** Based on the missing value distribution, you can make informed decisions about how to handle missing data. This might involve imputation (filling in missing values), removal of rows/columns with excessive missingness, or using techniques appropriate for the specific data and analysis goals.

9. df.isna().sum():

The code df.isna().sum() is a handy way to count missing values in each column of your pandas DataFrame, df. Here's a step-by-step breakdown of what it does:

- **`df.isna()`:** This part applies the isna() function from pandas to the DataFrame df. The isna() function identifies missing values (NaN or None by default) in each element of the DataFrame and returns a new DataFrame with Boolean values (True for missing values, False for valid values).
- **.sum():** This applies the sum() function element-wise across the rows (axis=0 by default) of the boolean DataFrame returned by df.isna(). Since True represents a missing value and False represents a valid value, summing across the rows essentially counts the number of True values (missing values) in each column.
- df.isna().sum() typically returns a Series containing the count of missing values for each column in the original DataFrame df. This Series acts like a dictionary, with column names as keys and the corresponding count of missing values as values.

10. Calculating the mean score :

- **Calculates Mean Score:**
    1. It calculates the average of three existing columns: math score, Science score, and English score (note the space at the end).
    2. The division by 3 computes the mean.
    3. The .round() function rounds the result to the nearest integer by default. Here, arguments are missing, so it rounds to the nearest integer.
- **Creates New Column:**
    1. It creates a new column named "mean" in the DataFrame df.
    2. This new column stores the calculated average scores for each student.
- **Displays DataFrame Head:**
    1. The df.head() method displays the first 5 rows of the modified DataFrame.
    2. This allows you to see the newly created "mean" column with the calculated average scores.

11. Intializing the passmarks:

- **Initializing Pass Mark:**

- passmark = 35 sets a global variable named passmark with a value of 35. This likely represents the minimum score required to pass a course.
- **Defining the Grade Function:**
- The def Grade(mean): function takes a single argument, mean, which presumably represents the average score for a student.
- The function uses a series of if statements to assign letter grades based on the following criteria:
    1. O: If the mean is greater than or equal to 95.
    2. A: If the mean is greater than or equal to 81.
    3. B: If the mean is greater than or equal to 71.
    4. C: If the mean is greater than or equal to 61.
    5. D: If the mean is greater than or equal to 51.
    6. E: If the mean is greater than or equal to 41.
    7. F: Otherwise (if the mean falls below 41).

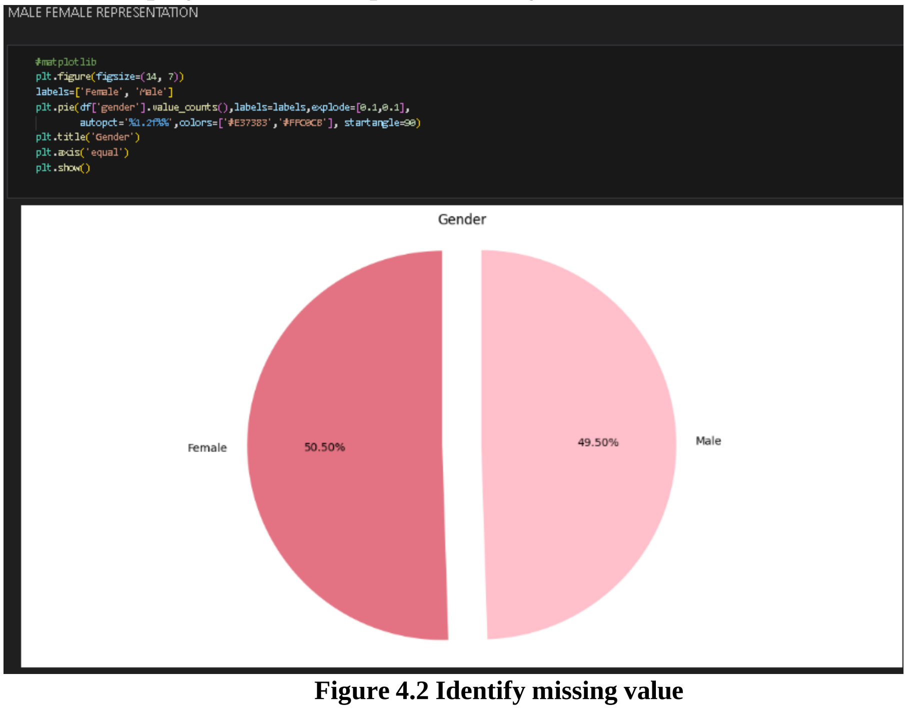
12. Adding the grade Column:

- df["grade"] = df.apply(lambda x : Grade(x["mean"]), axis=1) applies the Grade function to each row of the DataFrame df.
    1. lambda x : Grade(x["mean"]) is an anonymous function that extracts the value from the "mean" column within each row (x) and passes it to the Grade function.
    2. axis=1 specifies that the function is applied to each row (axis 0 would apply it to each column).
- This essentially calculates the grade based on the mean score for each student and adds a new column named "grade" to the DataFrame containing the letter grades.

13. Male female representation:

- plt.figure(figsize=(14, 7)) creates a new figure window for the pie chart.
- figsize=(14, 7) sets the width (14) and height (7) of the figure window in inches. This allows you to customize the size of your chart.

- plt.pie(df['gender'].value_counts(),labels=labels,explode=[0.1,0.1], autopct='%1.2f%%',colors=['#E37383','#FFC0CB'], startangle=90) creates the pie chart itself:
    1. df['gender'].value_counts(): This calculates the number of occurrences for each unique value in the gender column. It essentially counts the number of females and males in your data.
    2. labels=['Female', 'Male']: This defines the labels for each pie slice, corresponding to the female and male counts.
    3. explode=[0.1,0.1]: This separates the pie slices slightly by 0.1 units each, creating a 3D effect. You can adjust these values to control the separation.
    4. autopct='%1.2f%%': This specifies how percentages should be displayed on the pie chart. Here, it shows percentages with one decimal place and a percent sign (%).
    5. colors=['#E37383','#FFC0CB']: This assigns specific colors to the pie slices. You can choose different color codes based on your preference.
    6. startangle=90: This sets the starting angle for the pie chart. Here, it starts at 90 degrees, placing the "Female" slice at the top.
- plt.title('Gender') adds a title "Gender" to the pie chart, specifying the data it represents.
- plt.axis('equal') ensures the pie chart is drawn as a circle and avoids distortion.
- plt.show() displays the created pie chart on your screen.


MALE FEMALE REPRESENTATION

```
#matplotlib
plt.figure(figsize=(14, 7))
labels=['Female', 'Male']
plt.pie(df['gender'].value_counts(),labels=labels,explode=[0.1,0.1],
        autopct='%1.2f%%',colors=['#E37383','#FFC0CB'], startangle=90)
plt.title('Gender')
plt.axis('equal')
plt.show()
```

Figure 4.2 Identify missing value

14. Gender vs Grade graph:

- plt.figure(figsize=(10,5)) creates a new figure window for the countplot with a width of 10 and height of 5 inches.
- sns.set_context("talk",font_scale=1) sets the visual style context for the plot. "talk" is suitable for presentations as it uses a larger font size.
- sns.set_palette("pastel") sets the color palette for the plot to "pastel" colors, creating a softer look. These lines can be omitted if you prefer the default settings.
- ax=sns.countplot(y="grade",hue="gender",data=df,order=["O","A","B","C","D"," E","F"]) generates the countplot itself:
    1. y="grade": This specifies the variable on the y-axis, which is the "grade" column in your DataFrame. The countplot will show the number of students in each grade category.
    2. hue="gender": This introduces a hue variable, which is the "gender" column. The countplot will be colored by gender, allowing you to compare grade distribution between females and males.
    3. data=df: This indicates the data source, which is your DataFrame df.
    4. order=["O","A","B","C","D","E","F"]: This sets the order of the grade categories on the y-axis. Here, you've explicitly defined the order from best ("O") to worst ("F").
    5. ax.legend(loc='upper right',frameon=True) creates a legend in the upper right corner (loc='upper right') with a frame (frameon=True) to distinguish it from the plot. The legend explains the coloring based on gender.
- plt.title('Gender vs Grades', fontsize=18, fontweight='bold') sets the plot title to "Gender vs Grades" with a larger font size (18) and bold font weight for emphasis.
- ax.set(xlabel='COUNT',ylabel='GRADE') sets labels for the x-axis ("COUNT") and y-axis ("GRADE") to clarify what the plot represents.
- plt.show() displays the created countplot on your screen.

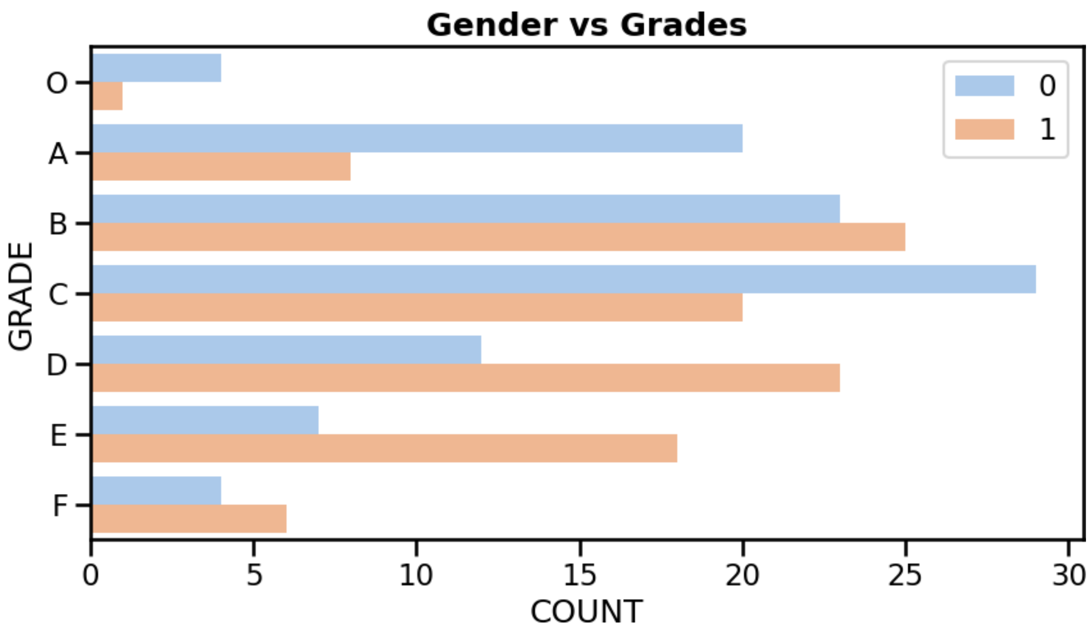


Figure 4.3 Gender vs Grades

15. Grade distribution with respect to gender:

- labels and values: These define the labels (grade categories) and their corresponding total counts.
- labels_gender and sizes_gender: These define labels with gender information (e.g., "F-O", "M-A") and their corresponding counts. You likely created this data by combining the "grade" and "gender" columns from your DataFrame.
- colors and colors_gender: These define color palettes for the two pie charts.
- The first plt.pie call creates the outer pie chart:
    1. values: This specifies the data points (total counts) for each grade category.
    2. labels: This sets the labels for each pie slice based on the grade categories.
    3. autopct='%1.1f%%': This displays percentages with one decimal place on the pie chart slices.
    4. pctdistance and labeldistance: These control the placement of percentages and labels relative to the pie center (outside in this case).
    5. colors: This assigns colors to the pie slices based on the colors list.
    6. startangle=90: This sets the starting angle for the pie chart.
    7. frame=True: This draws a frame around the pie chart.
    8. explode: This separates each slice slightly for a 3D effect.
    9. radius=12: This sets the radius of the outer pie chart.

- The second plt.pie call creates the inner pie chart:
    1. sizes_gender: This specifies the data points (counts for each gender-grade combination) for the inner pie chart slices.
    2. labels_gender: These labels likely indicate the gender ("M" or "F") followed by the grade ("O", "A", etc.).
    3. colors_gender: This assigns colors to the inner pie chart slices based on the colors_gender list.
    4. startangle=90: This sets the starting angle for the inner pie chart, aligning it with the outer one.
    5. explode: This separates each slice slightly for a 3D effect.
    6. radius=7: This sets the radius of the inner pie chart, making it smaller than the outer one.
- The code creates a circle object (centre_circle) with a black outline, white fill, and a specific radius (5) to cover the overlapping area between the two pie charts.
- It then adds this circle object (centre_circle) as an artist to the current plot using fig.gca().add_artist(centre_circle).
- plt.title sets the plot title with additional information about the gender representation within the pie slices.
- plt.axis('equal') ensures the pie charts are drawn as circles.
- plt.tight_layout() adjusts spacing to prevent overlapping elements.
- plt.show() displays the final plot.

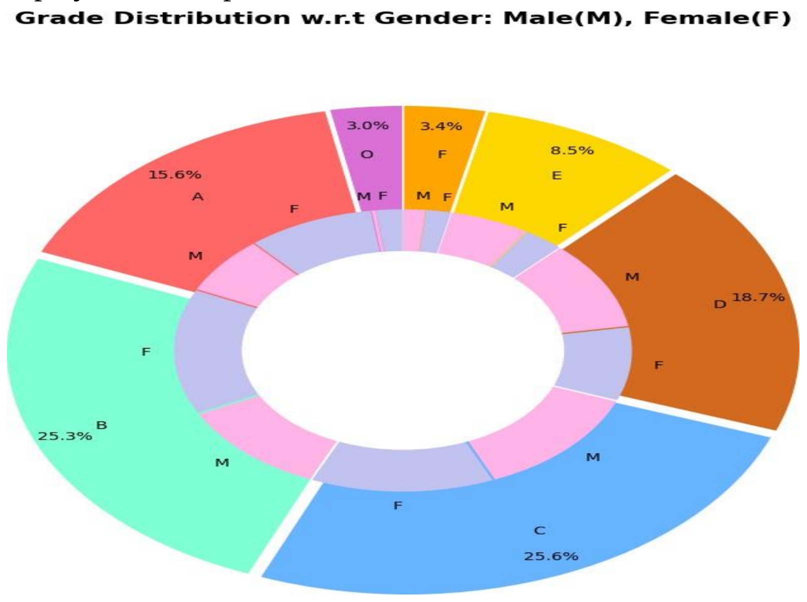


Figure 4.4 Grade distribution w.r.t male and female

16. Percentage vs test preparation:

- sns.set_context("talk",font_scale=0.5) sets the visual style context for the plot to "talk," which is suitable for presentations as it uses a smaller font size (0.5). You can adjust this value based on your needs.
- sns.set_palette("Pastel2") sets the color palette for the plot to "Pastel2" colors, creating a visually appealing look.
- sns.kdeplot(data=df, x="mean", hue="test preparation course", multiple="stack") generates a kernel density estimation (KDE) plot:
    1. data=df: This specifies the data source, which is your DataFrame df.
    2. x="mean": This indicates the variable on the x-axis, which is the "mean" column representing the average scores.
    3. hue="test preparation course": This introduces a hue variable, which is the "test preparation course" column. The plot will be colored by participation (1) or non-participation (0) in the course.
    4. multiple="stack": This stacks the KDE distributions for each category (course participation) on top of each other, allowing you to compare the density of scores for students who took the course versus those who didn't.
-
- plt.title('Percentage vs Test Preparation',fontsize=15, fontweight='bold') sets the plot title to "Percentage vs Test Preparation," clarifying what the plot represents. "Percentage" might not be the most accurate term here, as the y-axis typically represents probability density in KDE plots.plt.show() displays the created KDE plot on your screen.

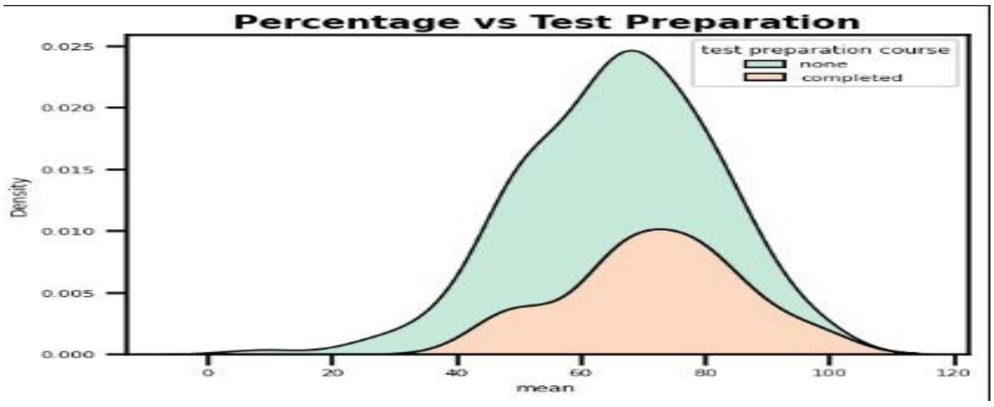


Figure 4.5 percentage vs test preparation

17. Parental education distribution vs gender:

- sns.catplot(x="parental level of education", y="mean", hue="gender", kind="violin", inner="stick", split=True, palette="pastel", data=df) creates the violin plot:
    1. x="parental level of education": This sets the categorical variable on the x-axis, which is the "parental level of education" column.
    2. y="mean": This sets the continuous variable on the y-axis, which is the "mean" column representing the average scores.
    3. hue="gender": This introduces a hue variable, which is the "gender" column. The violin plot will be colored by gender (female vs. male).
    4. kind="violin": This specifies the type of plot as a violin plot. Violin plots are useful for visualizing the distribution of a continuous variable (average scores here) across categories (parental education levels).
    5. inner="stick": This adds a vertical line within each violin representing the median score for each category and gender combination.
    6. split=True: This creates separate violin plots for each gender on the x-axis, allowing for easier comparison of score distributions.
    7. palette="pastel": This sets the color palette to "pastel" for a visually appealing look.
    8. data=df: This indicates the data source, which is your DataFrame df.
- plt.title('Parental Education Distribution vs Gender',fontsize=15, fontweight='bold') sets the plot title to clarify what the violin plot represents.
- plt.xticks(rotation=60) rotates the x-axis labels by 60 degrees to improve readability if there are many parental education levels.
- plt.show() displays the created violin plot on your screen.

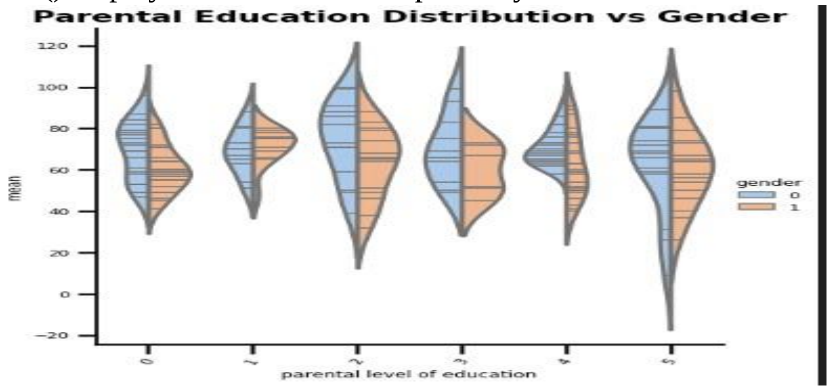
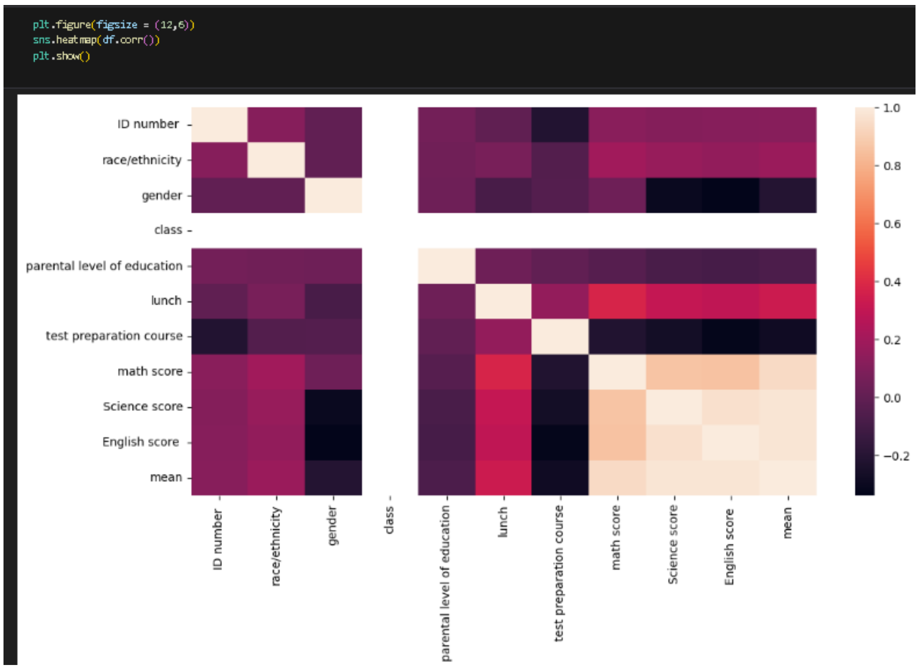


Figure 4.6 Parent education distribution vs gender

18. Heatmap:

- plt.figure(figsize = (12,6)) creates a new figure window for the heatmap with a width of 12 inches and a height of 6 inches. This allows you to control the overall size of the plot.df.corr(): This calculates the correlation matrix of your DataFrame df. The correlation matrix represents the pairwise correlation coefficients between all features (columns) in your data.
- sns.heatmap: This function takes the correlation matrix as input and visualizes it as a heatmap. Colors represent the correlation coefficients, with a color legend typically indicating the range of correlation values.
- plt.show() displays the created heatmap on your screen.



```
plt.figure(figsize = (12,6))
sns.heatmap(df.corr())
plt.show()
```

Figure 4.7 Heatmap

18.Website:

In order to effectively communicate the findings of our student performance analysis, we have developed a straightforward HTML website. This website visually presents the analysis results using graphs, making it easier to comprehend student performance trends and identify areas for improvement. This user-friendly interface allows viewers to quickly grasp the key insights derived from the analysis, aiding informed decision-making processes.

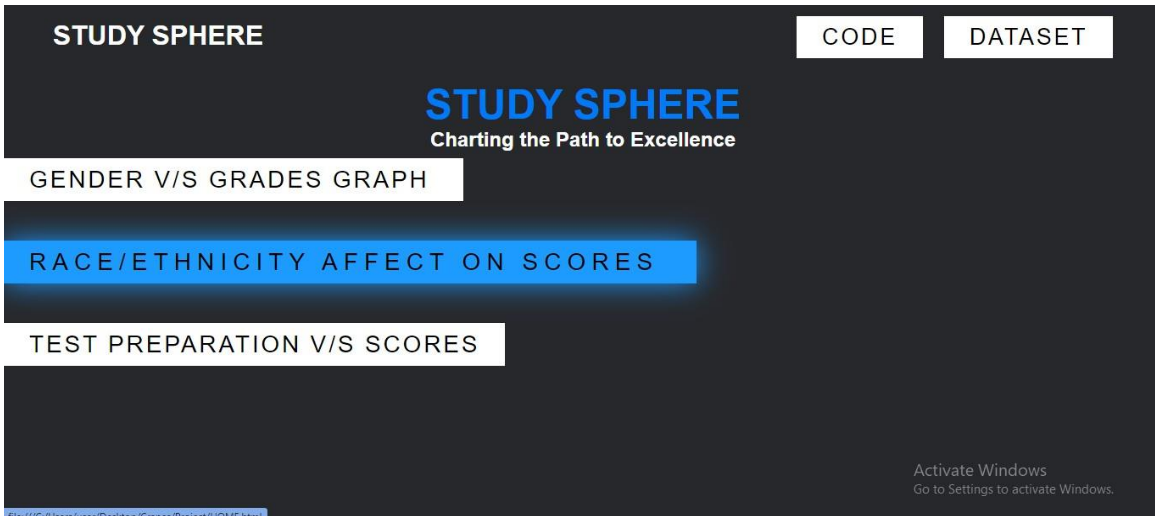

STUDY SPHERE
CODE
DATASET
STUDY SPHERE
Charting the Path to Excellence
GENDER V/S GRADES GRAPH
RACE/ETHNICITY AFFECT ON SCORES
TEST PREPARATION V/S SCORES
Activate Windows
Go to Settings to activate Windows.

Figure 4.8 Sample Website to represent the output

CONCLUSION

My data science with AIML internship culminated in a rewarding experience that equipped me with valuable skills for tackling real-world problems. I honed my abilities in data wrangling, analysis, and visualization using Python libraries like Pandas and Matplotlib. I gained practical experience with machine learning algorithms, likely working with decision trees or random forests to build models for student performance analysis. The internship not only solidified my understanding of data science fundamentals but also exposed me to the power of AIML techniques. Overall, this internship fostered my passion for data science and AIML, and I'm confident in my ability to leverage these skills to contribute meaningfully to future projects.The future of data science with AIML (Artificial Intelligence & Machine Learning) is glowing bright. The ever-increasing volume of data coupled with advancements in AI algorithms is fueling a perfect storm for growth. We can expect to see data science with AIML become even more pervasive across industries, from automating tasks and optimizing processes to personalizing experiences and unlocking entirely new possibilities. This powerful combination will be instrumental in tackling complex challenges in healthcare, finance, education, and beyond. As data continues to be the new oil, data scientists with expertise in AIML will be the engineers shaping a data-driven future.

REFERENCES

[1] https://www.cranesvarsity.com/, "Cranes Varsity Official Website", [Online]. Available: https://www.cranesvarsity.com/. [Accessed: 26-Dec-2023].

[2] Kaggle, "Kaggle Tutorial", [Online]. Available: https://www.kaggle.com/ [Accessed: 10-Nov-2023].

[3] geeksforgeeks, "Logistic Regression”, [Online]. Available: https://www.geeksforgeeks.org/understanding-logistic-regression [Accessed: 22-Nov-2023].

[4] Pandas, "DataFrame.to_numpy", [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_numpy.html [Accessed: 21-Nov-2023].