COAPPS PROJECT

on

STUDENT SPENDING PATTERN

Submitted in partial fulfillment for the completion of BACHELORS OF ENGINEERING., VIII Semester CSE(Artificial Intelligence & Data Science)

By

V. SRILATHA – 20ND1A54D6 (Team leader)

P. ARUNDATHI – 20ND1A5490

K. KEERTHI – 20ND1A5458



DEPARTMENT OF COMPUTER

SCIENCE AND ENGINEERING

ST. MARY’S WOMEN’S ENGINEERING COLLEGE

Approved by AICTE, New Delhi, Affiliated to JNTUK,

Permitted by Govt. of A.P ACCREDITED BY “NAAC”



CERTIFICATION

This is to certify that the Project entitled ”STUDENT SPENDING PATTERN“ submitted to COAPPS, in partial fulfillment the requirements for the completion of DATA SCIENCE TRAINING, VIII semester ARTIFICIAL INTELLLIGENCE AND DATA SCIENCE.

This is a record of original work done by

V. SRILATHA (20ND1A54D6)(Team leader),

K. KEERTHI (20ND1A5458),

P. ARUNDATHI (20ND1A5490),

during the period of study in the Department of CSE(AI&DS) in 4-2 Semester, STMWEC, Guntur, Budampadu, under my supervision and guidance.

Project Guide

MR. AMYJOY

CHENNAI

DECLARATION

We the undersigned solemnly declare that the project report is based on our own work carried out during the course of our study under the supervision of MR. AMYJOY, We certify that: The portrayal of the “Student Spending Pattern” is original and has been done by us under the general supervision of our supervisor. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad. We have followed the guidelines provided by the university in writing the report. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

V. SRILATHA – 20ND1A54D6(Team Leader)

K. KEERTHI – 20ND1A5458

P. ARUNDATHI – 20ND1A5490

ACKNOWLEDGEMENT

We would like to express our special thanks to our Guide MR. AMYJOY , who gave us the golden opportunity to do this project titled ‘Student Spending Pattern’ which helped us learn new skills regarding data science with python as well as web technologies. It also helped us apply the knowledge we have gained to practical use. We would also like to thank our parents, family and friends without whom this project could not have been completed.

ABSTRACT

This project aims to analyze the spending patterns of students using data science techniques. By leveraging data from various sources such as transaction records, surveys, and demographic information, we seek to understand how students allocate their finances across different categories such as education, food, entertainment, and more. The project involves data preprocessing, exploratory data analysis, and the application of machine learning algorithms to uncover insights into factors influencing spending behavior, such as income levels, geographic location, and student status. The findings from this study can provide valuable insights for financial planning, budgeting, and targeted marketing strategies tailored to the needs and preferences of the student demographic.

INTRODUCTION

1.1 INTRODUCTION

Understanding the spending patterns of students is crucial for various stakeholders, including educational institutions, policymakers, financial service providers, and students themselves. Student spending habits can provide valuable insights into their financial well-being, consumption behavior, and overall financial literacy. By analyzing student spending patterns, we can identify trends, challenges, and opportunities for promoting responsible financial management among the student population.

The aim of this project is to investigate and analyze the spending patterns of students using data science methodologies. With the advent of digital payments and online transactions, there exists a vast amount of data that can be leveraged to gain insights into how students allocate their finances. By applying data science techniques such as data collection, preprocessing, Exploratory Data Analysis (EDA), and machine learning, we can uncover hidden patterns and trends in student spending behavior.

This project will contribute to the existing body of knowledge on financial analysis and data-driven decision-making by focusing specifically on the unique spending habits of students. By gaining a deeper understanding of student spending patterns, we can develop targeted interventions and strategies to promote financial literacy, budgeting skills, and responsible spending habits among students. Additionally, the findings of this project can inform the design of educational programs, financial products, and policies aimed at supporting students in managing their finances effectively during their academic journey and beyond.

* 1. Import necessary Libraries

1. PANDAS

Pandas is a Python library used for data manipulation and analysis. It provides data structures and functions to efficiently work with structured data, such as tabular or time-series data. Pandas is particularly popular for tasks like data cleaning, transformation, exploration, and visualization. It offers powerful tools for reading and writing data in various formats (e.g., CSV, Excel, SQL databases), handling missing data, reshaping datasets, and performing statistical operations. Pandas is widely used in data science, machine learning, and quantitative finance due to its simplicity and flexibility in working with structured data.

1. NUMPY

NumPy is a Python library widely used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is essential for tasks such as numerical simulations, data analysis, machine learning, and scientific computing.

Key features of NumPy include:

1. Multidimensional array objects: NumPy arrays are homogeneous and can contain elements of the same data type, allowing for efficient storage and manipulation of large datasets.

2. Mathematical functions: NumPy provides a wide range of mathematical functions, such as trigonometric functions, statistical functions, linear algebra operations, and more, which can be applied directly to arrays without the need for explicit looping.

3. Broadcasting: NumPy's broadcasting feature allows arrays of different shapes to be combined in arithmetic operations, making it easier to perform element-wise computations on arrays with different dimensions.

4. Integration with other libraries: NumPy seamlessly integrates with other Python libraries such as SciPy (for scientific computing), Matplotlib (for data visualization), and pandas (for data analysis), forming a powerful ecosystem for numerical computing in Python.

Overall, NumPy is a fundamental library for numerical computing in Python, providing efficient data structures and mathematical functions that are essential for a wide range of scientific and engineering applications.

1. SEABORN

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn is particularly useful for visualizing relationships and distributions in datasets, making it a valuable tool for exploratory data analysis and communicating insights from data.

Key features of Seaborn include:

**1. High-level plotting functions:** Seaborn provides easy-to-use functions for creating complex plots with minimal code, allowing users to quickly generate informative visualizations without needing to delve into the intricacies of matplotlib.

**2. Statistical visualization:** Seaborn offers specialized functions for visualizing statistical relationships such as scatter plots, line plots, bar plots, box plots, violin plots, and regression plots, making it easier to explore and understand the underlying patterns in data.

**3. Integration with pandas:** Seaborn seamlessly integrates with pandas data structures, allowing users to pass Data Frame objects directly to plotting functions, making it convenient for working with structured data.

**4. Customization options:** Seaborn provides a wide range of customization options for fine-tuning the appearance of plots, including control over colors, styles, and annotations, enabling users to create visually appealing and publication-quality graphics.

Overall, Seaborn is a powerful and versatile library for data visualization in Python, offering a convenient interface for creating sophisticated statistical plots that facilitate the exploration and communication of insights from data.

1. MATPLOTLIB

Matplotlib is a comprehensive Python library for creating static, interactive, and animated visualizations. It is widely used for generating plots, charts, histograms, and other graphical representations of data. Matplotlib provides a high level of customization and flexibility, allowing users to create a wide range of plots to suit their specific needs.

Key features of Matplotlib include:

**1. Support for various plot types:** Matplotlib supports a wide variety of plot types, including line plots, scatter plots, bar plots, histograms, pie charts, contour plots, and more, making it suitable for visualizing different types of data.

**2. Fine-grained control over plot elements:** Matplotlib allows users to customize every aspect of their plots, including colors, line styles, marker styles, fonts, labels, annotations, and axis limits, enabling precise control over the appearance of the visualizations.

**3. Multi-platform compatibility:** Matplotlib works seamlessly across different platforms and operating systems, including Windows, macOS, and Linux, making it a versatile choice for generating plots in various environments.

**4. Integration with other libraries:** Matplotlib integrates well with other Python libraries such as NumPy, pandas, and SciPy, allowing users to seamlessly incorporate data from these libraries into their plots.

**5. Support for interactive and animated visualizations:** Matplotlib provides functionality for creating interactive plots with features like zooming, panning, and tooltips, as well as support for creating animated plots using the animation module.

Overall, Matplotlib is a powerful and versatile library for creating high-quality visualizations in Python, suitable for a wide range of applications in data analysis, scientific computing, engineering, and more.

1. SKLEARN

Scikit-learn, often abbreviated as sklearn, is a popular machine learning library for Python. It provides a simple and efficient set of tools for data mining and data analysis tasks. Scikit-learn is built on top of other scientific computing libraries like NumPy, SciPy, and matplotlib, making it seamlessly integrate with the Python data science ecosystem.

Key features of scikit-learn include:

**1. Simple and consistent API:** Scikit-learn offers a uniform interface for various machine learning algorithms, making it easy to experiment with different models without having to learn new syntax or APIs for each algorithm.

**2. Wide range of algorithms:** Scikit-learn includes implementations of various supervised and unsupervised learning algorithms, including linear and logistic regression, support vector machines (SVM), decision trees, random forests, k-nearest neighbors (KNN), clustering algorithms, dimensionality reduction techniques, and more.

**3. Data preprocessing and feature engineering:** Scikit-learn provides utilities for data preprocessing tasks such as feature scaling, normalization, imputation of missing values, encoding categorical variables, and feature selection, enabling users to prepare their data for modeling efficiently.

**4. Model evaluation and validation:** Scikit-learn offers tools for evaluating the performance of machine learning models through cross-validation, grid search for hyperparameter tuning, model selection, and metrics for classification, regression, and clustering tasks.

**5. Integration with other libraries:** Scikit-learn seamlessly integrates with other Python libraries such as pandas, allowing users to easily pass data from one library to another for preprocessing and modeling tasks.

Overall, scikit-learn is a versatile and user-friendly library that is widely used for building machine learning models in Python, suitable for both beginners and experienced practitioners in the field of data science and machine learning.

RAMDOM FOREST

Random Forest is a popular ensemble learning algorithm used for both classification and regression tasks in machine learning. It belongs to the family of decision tree-based algorithms and is known for its high performance and robustness.

Here's how Random Forest works:

**1.** **Decision Trees:** Random Forest is composed of multiple decision trees, where each tree is trained on a random subset of the training data and a random subset of features.

**2.** **Bootstrap Aggregating** **(Bagging):** Random Forest employs a technique called bagging, which involves creating multiple subsets of the training data through random sampling with replacement. Each decision tree in the Random Forest is trained on one of these subsets.

**3. Random Feature Selection:** In addition to using random subsets of the training data, Random Forest also uses random subsets of features for each tree. This helps to decorrelate the trees and reduces overfitting.

**4. Voting or Averaging:** For classification tasks, the prediction of a Random Forest is determined by a majority vote among the individual trees. For regression tasks, the predictions of all trees are averaged to obtain the final prediction.

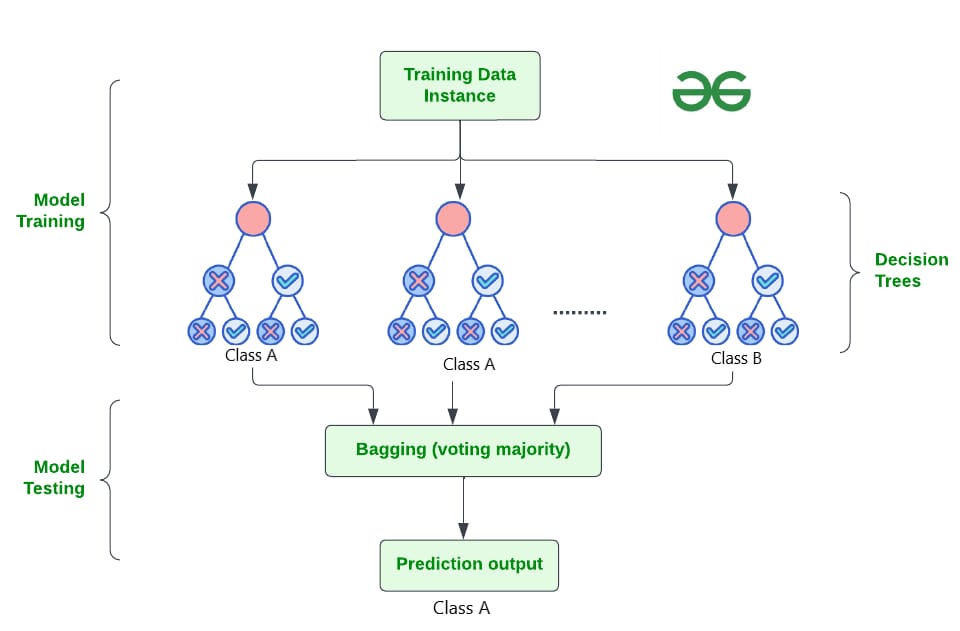
Random Forest offers several advantages:

**- High Performance:** Random Forest generally performs well on a wide range of datasets and is less prone to overfitting compared to individual decision trees.

**- Robustness:** It can handle noisy data and missing values effectively.

**- Feature Importance:** Random Forest can provide insights into feature importance, which can be useful for feature selection and understanding the underlying patterns in the data.

Overall, Random Forest is a versatile and powerful algorithm that is widely used in various machine learning applications, including classification, regression, and feature selection.

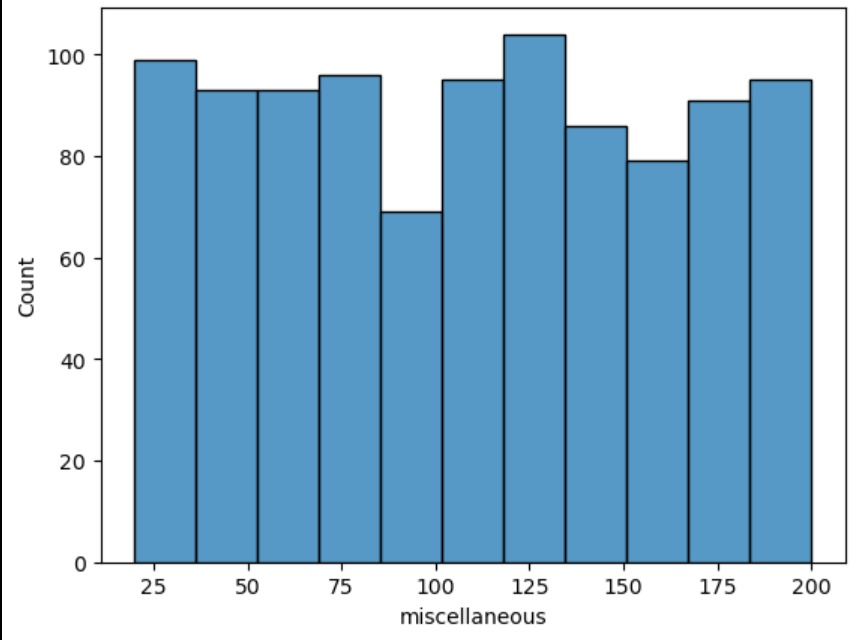


**RANDOM FOREST ALGORITHM**

HISTOGRAM

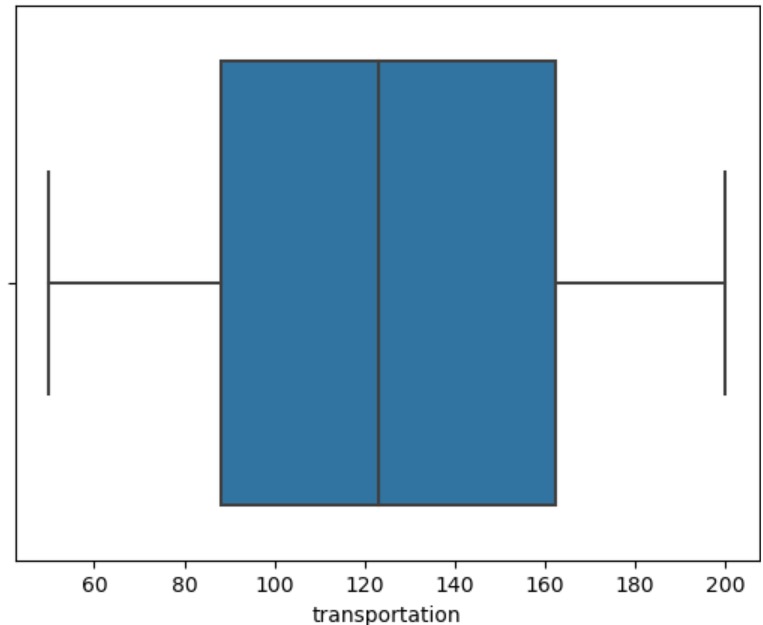
A histogram is a graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. A histogram is a type of bar chart that shows the number of data points that fall into each of several categories.

In Python, you can create a histogram using the **matplotlib.pyplot.hist()** function. This function takes two required arguments: the data to be plotted and the number of bins to use. The bins are the intervals into which the data will be divided.



BOXPLOT

A box plot, also known as a box-and-whisker plot or diagram, is a graph that summarizes a set of data by showing the distribution of the data, outliers, and other key statistics. Box plots are non-parametric, meaning they show variation in samples of a statistical population without assuming the distribution. The box plot's shape indicates how the data is distributed, and lines extending from the box indicate variability outside the upper and lower quartiles. The spacings in each subsection of the box plot indicate the degree of dispersion and skewness of the data.

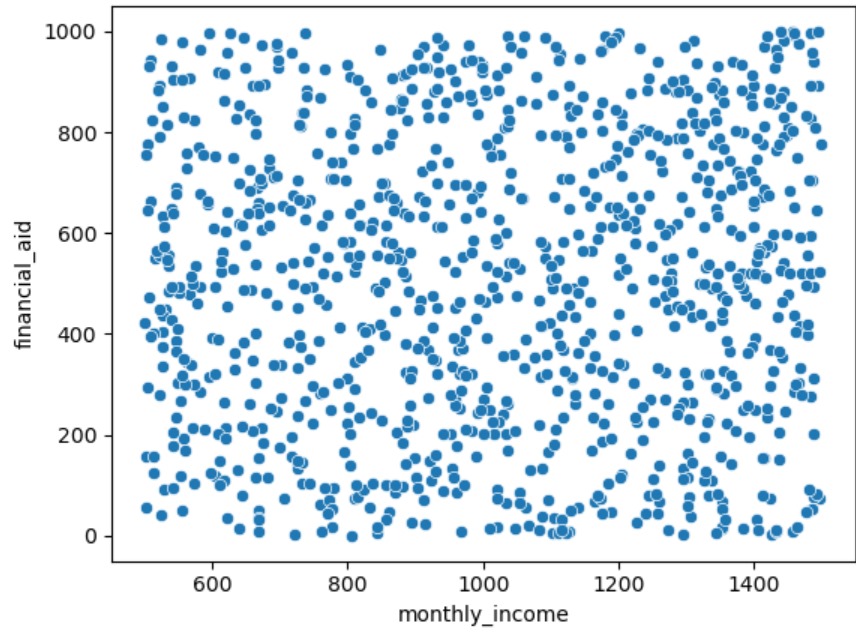


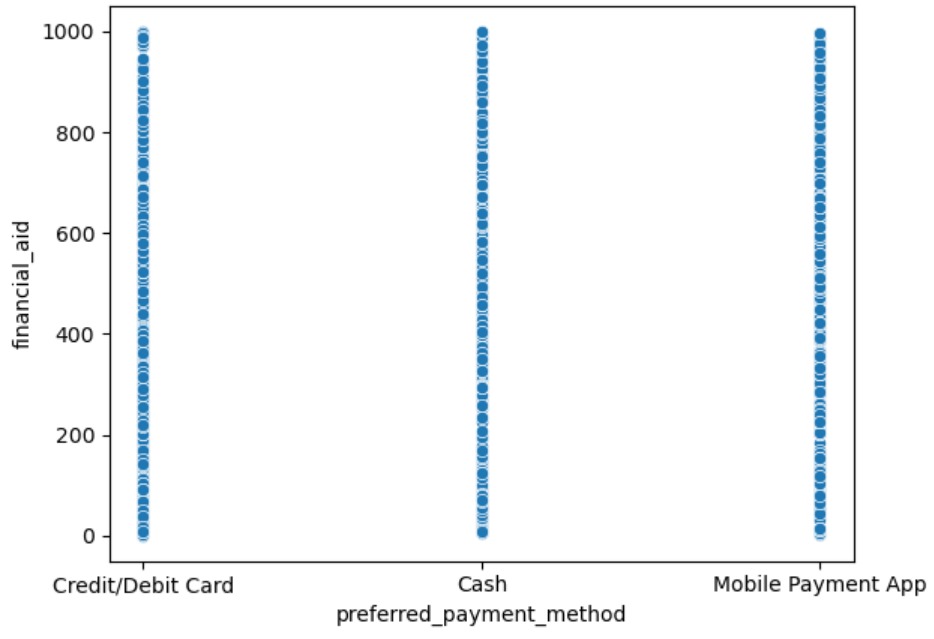
SCATTER PLOT

A scatter plot is a type of plot that displays the relationship between two variables. The points on the plot are called markers, and each marker represents a single data point. The position of the marker on the x-axis and y-axis corresponds to the values of the two variables for that data point.

Scatter plots can be used to identify patterns and trends in data. For example, a scatter plot of height vs. weight might show that there is a positive correlation between the two variables, meaning that taller people tend to weigh more.

To create a scatter plot in Python, you can use the matplotlib.pyplot.scatter() function. This function takes two arrays of data points as input, one for the x-axis and one for the y-axis. The function then plots each data point as a marker on the plot.





HEAT MAP

In data science, a heatmap is a grid of colored squares that shows the values of a main variable across two axis variables. The axis variables are divided into ranges, and each cell's color indicates the value of the main variable in the corresponding cell range. Heatmaps are a popular data visualization technique that uses color to represent different levels of data magnitude.

Heatmaps are used in various forms of analytics, but are most commonly used to show user behavior on specific web pages or webpage templates. Heatmaps can be used to:

* Show where users have clicked on a page
* Show how far users have scrolled down a page
* Identify relationships between two variables
* Help users quickly understand the most important or relevant data points

Heatmaps are essential in detecting what does or doesn't work on a website or page, and which parts and elements of a page users engage with. According to the Social Science Research Network, 65% of human beings are visual learners, and that is why visualizing data of any form makes so much more sense.

You can use Seaborn's heatmap function to create a heatmap. You can also customize heatmaps in Matplotlib by changing the color map, axis labels, title, adding a color bar, adjusting the size and aspect ratio, setting the minimum and maximum values, adding annotations, adjusting the cell size, masking certain cells, and adding borders.



STREAMLIT

Streamlit is an open-source Python library that enables data scientists and machine learning engineers to create interactive web applications in a fast and easy way. It is built on top of popular Python libraries such as NumPy, Pandas, and Matplotlib, and it can be used to create a wide variety of applications, including:

Data visualization dashboards, Machine learning model deployment applications, Interactive tutorials and demos, and Data science collaboration tools.

Streamlit applications are typically deployed on publicly accessible servers or cloud instances so that users can access them over the internet. However, it is also possible to deploy Streamlit applications locally for personal use or for sharing with a small group of people.

Here are some of the key features of Streamlit:

**Easy to use:**

Streamlit is designed to be easy to use, even for users with no prior experience in web development.

**Interactive:**

Streamlit applications are interactive, meaning that users can interact with the data and visualizations in real time.

**Fast:**

Streamlit applications are fast, even when working with large datasets.

**Deployable:**

Streamlit applications can be deployed on a variety of platforms, including Heroku, AWS, and Google Cloud Platform.

If you are a data scientist or machine learning engineer looking for a way to create interactive web applications, Streamlit is a great option to consider. It is easy to use, fast, and deployable, and it can be used to create a wide variety of applications.

Here are some of the benefits of using Streamlit:

**Increased productivity:**

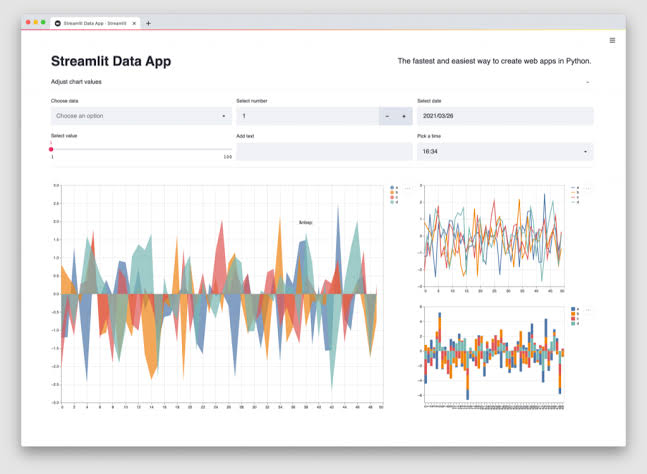
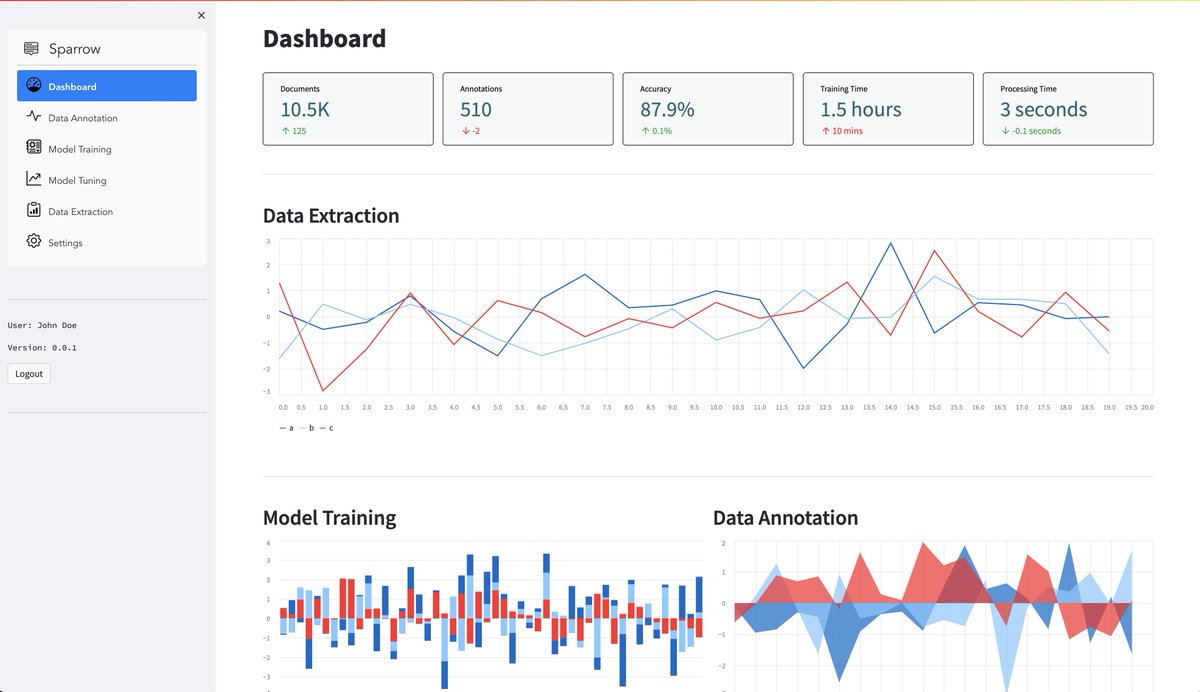
Streamlit can help data scientists and machine learning engineers to be more productive by enabling them to quickly create and deploy interactive web applications.

**Improved communication:**

Streamlit can help data scientists and machine learning engineers to communicate their work more effectively by enabling them to create interactive dashboards and reports.

**Greater collaboration:**

Streamlit can help data scientists and machine learning engineers to collaborate more effectively by enabling them to share interactive applications with each other.

CONCLUSION

Based on the analysis conducted using data science techniques, several key conclusions can be drawn from the student spending pattern project:

**1. Category Allocation:** The analysis reveals that a significant portion of students' spending is allocated towards essentials such as housing, food, and transportation, highlighting the importance of these categories in their budget.

**2. Spending Trends:** Through time-series analysis, it is observed that certain spending categories exhibit seasonal fluctuations or trends, indicating potential factors influencing students' spending behavior.

**3. Correlation Analysis:** By examining correlations between different spending categories, insights are gained into potential relationships and dependencies among expenses, which can inform budgeting strategies and financial planning.

**4. Predictive Modeling:** Utilizing predictive models, future spending patterns can be forecasted with a degree of accuracy, providing valuable insights for financial planning and decision-making.

**5. Recommendations:** Based on the findings, recommendations can be made to students on optimizing their spending, identifying areas for potential savings, and adopting effective budgeting practices to achieve financial goals.

Overall, the application of data science methodologies has enabled a comprehensive understanding of student spending patterns, offering valuable insights for both individuals and institutions aiming to improve financial literacy and management among students.