# Project Title: Customer Spending Limit Prediction

# Team Members:

* Vishnu A[Leader]
* Vijayasarathy R [Member]
* Jeya Aravinth S [Member]
* Sundramahalingam P [Member]
* Vignesh Prabhu M [Member]

# Abstract:

The retail company's marketing team aims to create a targeted marketing plan with a clear return on investment (ROI) through a machine learning application developed by their data science team. This application is designed to predict customers' spending limits based on their earnings and earning potential. In a meeting with the company's management, it was proposed that the machine learning model should be versatile, allowing users to upload training data and select features via a user-friendly interface. Additionally, users should be able to upload and preview test data for model evaluation.

To enhance the understanding of the model's outcomes, the MLOps system will incorporate an Explanations AI functionality, aiding business users in interpreting the results. Furthermore, the application will include visual data analysis features to simplify and clarify the insights generated by the model. This holistic approach will enable the retail company to develop effective marketing strategies and demonstrate the ROI of their marketing efforts.

# Project Overview:

**Customer Spending Limit Prediction Web Application**

**Project Description:**

The Customer Spending Limit Prediction Web Application is a machine learning-driven tool designed to assist a retail company in predicting customers' spending limits based on their earnings and earning potential. This project combines web development using Flask, machine learning with scikit-learn, and data handling with pandas**.**

**Components:**

**Web Interface (index.html):**

* This HTML template serves as the user interface for the application.
* Users can upload a CSV file containing customer data (with 'earnings' and 'earning\_potential' columns) for prediction.
* The interface is styled using Bootstrap for a user-friendly look and feel.
* Machine Learning Model (model.py):

A Linear Regression model is created and trained using historical customer data.

The trained model is saved as a joblib (.pkl) file for later use.

**Flask Web Application (app.py):**

* A Flask web application is created to handle user interactions and serve predictions.
* Users are presented with the home page where they can upload data files for prediction.
* Upon file submission, the web app reads the CSV file and uses the trained model to make predictions.
* Predictions are added to the dataset, and the results are displayed on the web page.
* Error handling is in place to handle invalid file uploads.

**Workflow:**

* Users visit the home page of the web application.
* Users upload a CSV file with customer data that includes 'earnings' and 'earning\_potential' columns.
* The Flask app reads the uploaded data, makes predictions using the pre-trained Linear Regression model, and adds the predictions to the dataset.
* The dataset, now including predicted spending limits, is converted into an HTML table.
* The results are displayed on the web page, allowing users to review and analyze the predicted spending limits.

**Purpose:**

The project's main objective is to provide a user-friendly tool for the retail company's marketing team to predict customer spending limits based on income-related features. By combining machine learning and web development, the application empowers users to make data-driven decisions and develop targeted marketing strategies. This, in turn, allows the company to optimize their marketing efforts and achieve a higher return on investment (ROI).

**Technologies Used:**

HTML for front-end UI.

Bootstrap for styling the UI.

Python for back-end development.

Flask for creating the web application.

pandas for data handling.

scikit-learn for building and training the Linear Regression model.

joblib for saving and loading the trained model.

# Data Collection and Preprocessing:

The provided dataset appears to contain information about customers, including their "earnings," "earning potential," and "spending limit." Here's how data collection and preprocessing could be described for this dataset:

**Data Source:** The data appears to be collected from various customers, and it's available in a structured format, likely in a CSV file.

**Data Fields:**

"earnings" represents the customer's current earnings.

"earning\_potential" indicates the customer's earning potential.

"spending\_limit" is the spending limit for each customer.

**Data Quality Check:**

During data collection, it's important to perform a quality check to ensure there are no missing values.

We don't have missing values or categorical variables, so the preprocessing steps are relatively straightforward. It's important to adjust the preprocessing steps according to the nature of your dataset and the requirements of the machine learning model you plan to build.

# Model Architecture:

Model Architecture: Simple Linear Regression

**Model Type:** Linear Regression

Linear regression is a type of regression analysis used to predict a continuous target variable (spending limit in this case) based on one or more independent input features (earnings and earning potential).

**Input Features:**

earnings: Represents the customer's current earnings.

earning\_potential: Indicates the customer's earning potential.

**Target Variable:**

spending\_limit: The spending limit for each customer.

**Model Training:**

The linear regression model is initialized and trained using the provided dataset.

The model learns the coefficients (weights) for the input features to create a linear equation to predict the target variable.

**Model Saving:**

After training, the model is saved as a .pkl file using the joblib library for future use.

# Training Process:

The training process for the linear regression model used to predict customer spending limits based on earnings and earning potential consists of several steps. Here's a high-level overview of the training process:

**Data Loading:**

Load the dataset containing the customer data, including 'earnings,' 'earning\_potential,' and 'spending\_limit.'

**Data Preprocessing:**

Check and handle missing values: Ensure that there are no missing values in the dataset. Handle missing values through imputation or removal if necessary.

**Feature Selection:** Decide which features to include in the training data. In this case, 'earnings' and 'earning\_potential' are used.

**Data Splitting:** Split the dataset into a training set and a testing set for model evaluation. The training set is used to train the model, and the testing set is used to assess its performance.

Feature Scaling (if necessary):

Normalize or scale the features if required. In the case of linear regression, scaling may not be necessary, but it can help with some machine learning algorithms.

**Model Training:**

Fit the model to the training data to learn the coefficients (weights) for the linear equation that predicts the target variable (spending\_limit)

**Model Evaluation (optional):**

If you have a separate testing dataset, evaluate the model's performance on that dataset using relevant metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared (R²).

**Model Saving:**

Once the model is trained, save it as a .pkl file using a library like joblib for future use. This allows you to load the model and make predictions without retraining.

# Evaluation Metrics:

**R-squared (R²) or Coefficient of Determination:**

* Meaning: R² measures the proportion of the variance in the actual spending limits that is explained by the model's predictions. It ranges from 0 to 1.
* Interpretation: An R² value close to 1 suggests that the model explains a large portion of the variance in the spending limits. Conversely, an R² value close to 0 indicates that the model doesn't explain much variance. However, R² alone may not tell you if the model is clinically significant.
* It's important to note that R² can be misleading when the dataset is small or the relationship is non-linear. Additionally, it can't tell you if the coefficients are individually significant or if the model is overfitting.

# Results and Discussion:

The Customer Spending Limit Prediction Application is designed to predict customer spending limits based on their earnings and earning potential. Users can upload a CSV file containing customer data, and the application utilizes a trained linear regression model to make predictions. Here are the results and a discussion of the application:

* User-Friendly Interface: The web application provides an easy-to-use interface for users to upload their data and obtain predictions. The interface is designed with Bootstrap for a clean and intuitive look.
* Data Processing: The application effectively processes the uploaded data, assuming that it adheres to the expected format with 'earnings' and 'earning\_potential' columns.
* Prediction: The linear regression model, which has been trained using historical customer data, makes predictions for spending limits. It adds these predictions to the dataset and displays them to the user.
* Results Presentation: The application presents the results in a user-friendly format, showing the predicted spending limits alongside the original data. This makes it easy for business users to understand and analyze the predictions.
* Model Accuracy: The accuracy of the model depends on the quality of the data and the model's capacity to capture the underlying patterns in the data. The evaluation metrics (MAE, RMSE, R², etc.) are critical in assessing the model's performance. A lower MAE and RMSE and a higher R² value suggest better accuracy and a stronger relationship between the input features and spending limits.

**Discussion:**

Interpretability: Linear regression is a straightforward and interpretable model, making it suitable for understanding the relationships between earnings, earning potential, and spending limits. Users can easily grasp how changes in earnings and earning potential impact spending limits.

* Data Quality: The accuracy of predictions is highly dependent on the quality and representativeness of the dataset used for training. The application should include data quality checks and preprocessing steps to handle missing values or outliers.
* Model Performance: The choice of the linear regression model is just one approach. Depending on the dataset and its characteristics, other machine learning models, such as decision trees or random forests, might provide better performance. Users should explore different models to find the most suitable one.
* Evaluation Metrics: The use of appropriate evaluation metrics (e.g., MAE, RMSE, R²) helps in assessing the model's performance and ensuring that the predictions are reliable. Business users should be aware of these metrics to understand the model's accuracy.
* Model Updates: Over time, the model may require updates to adapt to changes in customer behavior and spending patterns. Regular model retraining is recommended to maintain prediction accuracy.
* Additional Features: To enhance the application, consider adding features like data visualization and user guidance for interpreting the results. Visualizations can help users understand the data and model outcomes more intuitively.
* In summary, the Customer Spending Limit Prediction Application provides a valuable tool for making data-driven decisions in retail marketing. It offers a user-friendly interface, accurate predictions, and the potential for further improvements in data quality, model choice, and interpretability. As the retail company continues to use and refine the application, it can gain insights that drive more effective marketing strategies and improved return on investment..

# Deployment:

**Deployment Steps:**

Deepsphere.AI

**Web Framework:**

We use the Flask web framework for deploying the machine learning model. Flask is a lightweight and versatile framework for building web applications, and it is well-suited for integrating machine learning models into web-based applications.

**API Endpoints:**

Create two main API endpoints for the application:

/: The root endpoint is the home page of the application where users can interact with the user interface to upload data and receive predictions.

/predict: This endpoint handles the prediction process when users submit data for prediction. It reads the uploaded data, makes predictions using the trained model, and returns the results.

**User Interface Components:**

The user interface components are built using HTML templates. Here's how they are implemented:

**HTML Template (index.html):** This template serves as the user interface for the application. It includes components such as a file upload form, buttons, and result display areas. It's designed using Bootstrap for a user-friendly appearance.

**Form Submission**: Users can upload a CSV file containing customer data with 'earnings' and 'earning\_potential' columns through the file input form.

**Predict Button:** Users trigger the prediction process by clicking the "Upload and Predict" button.

**Results Display:** After processing the uploaded data and making predictions, the results are displayed in a user-friendly table format on the same page.

**Data Processing and Model Loading:**

Within the Flask application, the data processing and model loading are performed in the /predict endpoint. The application reads the uploaded data, prepares it for prediction, loads the pre-trained linear regression model (previously saved as 'model.pkl'), and uses the model to make predictions.

**Error Handling:**

The application includes error handling to ensure that the uploaded data is correctly formatted and that predictions are made without errors. It can handle exceptions such as missing files or data format issues and provide informative error messages to the user.

**Instructions for Running the Project:**

**Prerequisites:**

**Install Python:** Make sure you have Python installed on your system. You can download it from the official Python website if it's not already installed.

**Install VS Code:** Install Visual Studio Code if you haven't already. You can download it from the official website.

**Project Setup:**

Open VS Code and use the "Open Folder" option to select the folder where Create

Create a Virtual Environment:

Open a terminal in VS Code by going to "Terminal" > "New Terminal."

Create a virtual environment in your project directory by running the following command:

bash

Copy code

python -m venv venv

Activate the Virtual Environment:

On Windows:

bash

Copy code

.\venv\Scripts\activate

On macOS and Linux:

bash

Copy code

source venv/bin/activate

You should see the virtual environment's name in the terminal prompt, indicating that the environment is active.

**Install Required Dependencies:**

With the virtual environment active, you can install the project's dependencies by running:

bash

Copy code

pip install -r requirements.txt

Run the Flask Application:

Run the Flask Application:

Ensure you are in the project directory with the virtual environment activated.

Run the Flask application by executing the following command:

bash

Copy code

flask run

You should see output indicating that the Flask development server is running.

**Access the Application:**

Open your web browser and go to http://127.0.0.1:5000/ to access the Customer Spending Limit Prediction web application. This is the default address and port used by the Flask development server.

**Usage:**

In your web browser, you can use the web interface to upload a CSV file containing customer data. The application will predict spending limits based on the provided data and display the results.

**Shutdown:**

To stop the Flask application, you can press Ctrl+C in the terminal where it is running. This will stop the development server.

By following these steps, you can set up and run the Customer Spending Limit Prediction project in Visual Studio Code with a virtual environment. This setup keeps your project dependencies isolated and allows you to develop and run the application efficiently.

# Code Snippets:

# import pandas as pd

# from sklearn.linear\_model import LinearRegression

# import joblib

# # Load your dataset

# data = pd.read\_csv("data.csv")

# # Define your features (X) and target variable (y)

# X = data[['earnings', 'earning\_potential']]

# y = data['spending\_limit']

# # Initialize and train a Linear Regression model

# model = LinearRegression()

# model.fit(X, y)

# # Save the trained model as a .pkl file

# joblib.dump(model, 'model.pkl')

# Conclusion:

The Hackathon project, "Customer Spending Limit Prediction," has achieved several milestones and demonstrated the capabilities of machine learning in retail marketing. Here's a summary of the project's achievements, lessons learned, and potential future improvements:

Achievements:

Predictive Model: The project successfully implemented a predictive model that estimates customer spending limits based on their earnings and earning potential. The model leverages linear regression, a simple and interpretable technique.

User-Friendly Interface: A user-friendly web interface was created using Flask and HTML templates. Users can easily upload their data, make predictions, and visualize the results.

Evaluation Metrics: The project incorporated evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R²) to assess model performance and ensure prediction accuracy.

Data Processing: The project outlined data collection and preprocessing steps, emphasizing the importance of data quality and feature engineering.

Deployment: The project provided guidelines for deploying the model using Flask and a virtual environment, making it accessible to users.

Lessons Learned:

Data Quality Matters: Data quality and preprocessing are crucial for model accuracy. Handling missing values and outliers is essential.

Model Selection: While linear regression is a good starting point, more complex models may offer improved predictive performance. The choice of model should be tailored to the data.

User Interface: A user-friendly interface enhances the application's usability and makes it accessible to non-technical users.

Continuous Improvement: Models need periodic updates to adapt to changing customer behavior and spending patterns. Regular retraining is necessary.

Potential Future Improvements:

Feature Engineering: Explore additional features that may impact spending limits, such as customer demographics, historical spending behavior, or seasonal trends.

Model Selection: Consider more advanced machine learning models, like decision trees or neural networks, to capture non-linear relationships in the data.

Visualizations: Enhance the user interface with data visualizations to provide users with a more intuitive understanding of the data and model outcomes.

Data Security: Implement data security measures to protect customer data and ensure compliance with privacy regulations.

Scaling: Consider scaling the application to handle a larger user base and provide robust performance under high traffic conditions.

Feedback Mechanism: Incorporate a feedback mechanism that allows users to provide insights and report issues for ongoing model improvement.

Real-time Predictions: Explore the possibility of making real-time predictions, allowing users to input data directly via the interface.

In conclusion, the Customer Spending Limit Prediction project serves as a foundation for leveraging machine learning in retail marketing. It has demonstrated the importance of data quality, model choice, and a user-friendly interface. Continuous improvement and adaptation to changing customer behavior will be key to its long-term success.

**References:**

Machine Learning and Data Science Tutorials:

Online platforms like Kaggle, Coursera, edX, and Udacity offer numerous tutorials and courses on machine learning and data science, including those focused on regression analysis and predictive modeling.

Machine Learning Libraries and Frameworks:

Official documentation and tutorials for popular machine learning libraries like scikit-learn and TensorFlow can provide guidance on model development, training, and evaluation.

Flask Web Development Tutorials:

Resources on Flask web development, including official documentation, tutorials, and online courses, can assist in building the web interface for deploying the model.

Data Preprocessing and Feature Engineering Guides:

Online articles and books on data preprocessing and feature engineering techniques can help improve the quality and relevance of the data used for prediction.

Evaluation Metrics Documentation:

Documentation for machine learning evaluation metrics like MAE, RMSE, and R² can help in understanding how to assess model performance.