**CLV** – Customer Lifetime Value

* The necessary libraries are imported: **pandas**, **numpy**, **train\_test\_split** from **sklearn.model\_selection**, **LinearRegression** from **sklearn.linear\_model**, **mean\_squared\_error** and **r2\_score** from **sklearn.metrics**.
* **df** is assigned the data loaded from an Excel file named **customer\_lifetime\_value.xlsx** using **pd.read\_excel()**.
* The feature variables (**X**) and target variable (**y**) are extracted from **df**.
* The data is split into training and testing sets using **train\_test\_split()**, with 80% for training and 20% for testing.
* An instance of the linear regression model (**LinearRegression()**) is created and assigned to **model**.
* The model is trained using the training data (**X\_train** and **y\_train**) using the **fit()** method.
* The model is used to make predictions on the testing data (**X\_test**), and the predicted values are stored in **y\_pred**.
* The root mean squared error (RMSE) and r-squared (R2) value are calculated using **mean\_squared\_error()** and **r2\_score()**, respectively, to evaluate the model's performance.
* The RMSE and R2 values are printed to the console.
* A sample set of feature values is defined in **sample\_values** for which the CLV will be predicted.
* The model is used to predict the CLV for the sample values, and the predicted CLV value is stored in **sample\_prediction**.
* The predicted CLV value for the sample values is printed to the console.

**Unsupervised Segmentation ML Model** -

* The necessary libraries are imported: **pandas**, **StandardScaler** from **sklearn.preprocessing**, **PCA** from **sklearn.decomposition**, **DBSCAN** from **sklearn.cluster**, and **matplotlib.pyplot** for plotting.
* An Excel file named **Cust\_segmentation\_and\_Recomendation.xlsx** is loaded using **pd.ExcelFile()** and assigned to the variable **xls**.
* The first sheet named **'Sheet1'** from the Excel file is parsed and assigned to the DataFrame **df**.
* Specific numerical columns from **df** are selected for clustering and stored in **df\_clustering**.
* A loop is used to convert numeric string values in each column of **df\_clustering** to float values.
* Specific categorical columns from **df** are selected for one-hot encoding and stored in **df\_categorical**.
* One-hot encoding is performed on **df\_categorical** using **pd.get\_dummies()** and the result is assigned to **df\_encoded**.
* The encoded categorical data in **df\_encoded** is combined with the numerical data in **df\_clustering** along the column axis using **pd.concat()** and assigned to **df\_combined**.
* Any missing values (NaNs) in **df\_combined** are filled with zeros.
* An instance of **StandardScaler()** is created and assigned to **scaler**. Then, the values in **df\_combined** are normalized using **scaler.fit\_transform()** and assigned to **df\_normalized**.
* An instance of **PCA()** is created with **n\_components=2**. Then, PCA is applied to **df\_normalized** using **pca.fit\_transform()** and assigned to **df\_pca**.
* An instance of **DBSCAN()** is created with specific parameters. Then, DBSCAN clustering is applied to **df\_pca** using **dbscan.fit\_predict()** and the cluster labels are assigned to **clusters**.
* The clusters are plotted using **plt.scatter()** with appropriate labels and color mapping.
* A new column named **'Cluster'** is added to the original DataFrame **df** and assigned the cluster labels.
* The DataFrame **df** is printed to the console, including the cluster labels assigned to each data point.

**Supervised Segmentation (Classification) ML Model Building** -

* The necessary libraries are imported: **numpy**, **pandas**, **train\_test\_split** from **sklearn.model\_selection**, **StandardScaler** from **sklearn.preprocessing**, **RandomForestClassifier** from **sklearn.ensemble**, **MultiOutputClassifier** from **sklearn.multioutput**, **confusion\_matrix** from **sklearn.metrics**, **TfidfVectorizer** from **sklearn.feature\_extraction.text**, and **accuracy\_score** from **sklearn.metrics**.
* An Excel file named **Cust\_segmentation\_and\_Recomendation.xlsx** is read using **pd.read\_excel()** and assigned to the DataFrame **df**.
* Specific columns in the DataFrame **df** are converted to the object or string data type using **astype(object)** or **astype(str)**.
* A new DataFrame named **df2** is created by selecting specific columns from **df**.
* A function named **custom\_tokenizer()** is defined to split text and value using regular expressions.
* Several instances of **TfidfVectorizer()** are created and applied to different columns in **df2** to generate TF-IDF matrices.
* The TF-IDF matrices generated for different columns are concatenated into a single DataFrame named **tfidf\_matrix**.
* The data is split into training and testing sets using **train\_test\_split()**. The TF-IDF matrix is assigned to **X\_train** and **X\_test**, while the target variables are assigned to **y\_train** and **y\_test**.
* An instance of **RandomForestClassifier()** is created with specific parameters and assigned to **classifier**. Then, a **MultiOutputClassifier** is created with **classifier** as the base estimator and assigned to **multioutput\_classifier**. The model is trained on the training set using **multioutput\_classifier.fit()**.
* Predictions are made on the testing set using **multioutput\_classifier.predict()** and stored in **y\_pred**.
* Accuracy scores for each output column are calculated using **accuracy\_score()** by comparing the predicted values **y\_pred** with the actual values **y\_test**. The accuracy scores are stored in **accuracies**, and the mean accuracy is computed and stored in **mean\_accuracy**.
* The accuracy score for each output column and the mean accuracy are printed to the console.

**Recommendation ML Model Building** -

* The necessary libraries are imported: **pandas**, **TfidfVectorizer** from **sklearn.feature\_extraction.text**, and **cosine\_similarity** from **sklearn.metrics.pairwise**.
* An Excel file named **Cust\_segmentation\_and\_Recomendation\_with\_Recommendations.xlsx** is loaded using **pd.ExcelFile()** and assigned to the variable **xls**. The names of all the sheets in the Excel file are stored in **sheet\_names**.
* The first sheet (index 0) from the Excel file **xls** is read using **pd.read\_excel()** and assigned to the DataFrame **df**.
* A list named **customer\_profile\_columns** is created, which contains column names representing customer profile information.
* A loop iterates over each column name in **customer\_profile\_columns** and converts the corresponding column in the DataFrame **df** to string data type using **astype(str)**.
* A new column named **'customer\_profile'** is created in **df**. It combines the values from the columns specified in **customer\_profile\_columns** into a single string for each customer.
* An instance of **TfidfVectorizer()** is created and assigned to **vectorizer**. Then, the vectorizer is fitted to the **'customer\_profile'** column in **df** using **vectorizer.fit\_transform()**, which converts the text data into TF-IDF (Term Frequency-Inverse Document Frequency) vectors. The resulting TF-IDF matrix is stored in **tfidf\_matrix**.
* The cosine similarities between the TF-IDF vectors in **tfidf\_matrix** are computed using **cosine\_similarity()**, and the resulting cosine similarity matrix is assigned to **cosine\_similarities**.
* A function named **recommend\_services()** is defined, which takes the row number of a customer and the top-n most similar customers to consider as inputs. This function recommends services based on the similarity scores between the customer and other customers.
* The **recommend\_services()** function is called with **row\_num=0** to get recommendations for the first row (customer) in the DataFrame, and the recommendations are stored in the variable **recommendations**.
* Finally, the recommendations are printed to the console.

Project Plan -

**Project Plan: Customer Segmentation and Service Recommendation Model**

I. **Project Introduction**

The project at hand involves developing a machine learning model to segment customers and recommend appropriate banking services based on various transactional and demographic parameters. The data for this project is sourced from Hadoop tables, specifically the 'Customer Master', 'Account Master', and 'Transaction Pool'.

II. **Project Objectives**

* To create an accurate customer segmentation model based on customer value, frequency of transactions, type of transactions, profession, and type of investor.
* To build a recommendation system that suggests relevant banking services ('Loan Recommendation', 'Investment Recommendation', 'Credit Card Recommendation', 'Insurance Recommendation') based on the customer segmentation.

**Data Preparation**

* **Data Extraction**: Implementing a robust method for extracting relevant data from the Hadoop tables, which includes 'Customer Master', 'Account Master', and 'Transaction Pool'.
* **Data Cleansing**: Developing procedures to cleanse and normalize the data, making it suitable for the modelling process. This includes handling missing values, outliers, and duplicate data.
* **Feature Engineering**: Delivering a robust feature engineering strategy which includes selecting, transforming, and creating features from raw data that are most suitable for machine learning models, we have used **TF-IDF Vectorization** technique here .
* **Categorical Representation**: Establishing a system to categorize monthly transactions into meaningful categories like 'Shopping', 'Investments', etc., for a more detailed customer behaviour analysis.

**Customer Segmentation**

* **Model Development**: Designing and deploying a Multioutput Classifier that segments customers based on various factors, such as 'Customer\_Value', 'Frequency\_of\_Transactions', 'Type\_of\_Transactions', 'Profession', and 'Type\_of\_Investor'.
* **Segmentation Output**: Creating a 'Segmentation' table in Hadoop to store the output of the segmentation model. The table will contain categorized customer data, forming the basis for targeted recommendations.

**Service Recommendation Model**

* **Model Development**: Developing a machine learning model that utilizes the outputs of the customer segmentation model to recommend banking services to customers.
* **Recommendation Categories**: Designing the model to suggest specific services such as 'Loan Recommendation', 'Investment Recommendation', 'Credit Card Recommendation', 'Insurance Recommendation' based on the customer's segmented category.
* **Recommendation Output**: Producing a detailed report or table of recommended services for each customer, aiding the bank's sales and marketing teams in targeting customers with personalized offers.

**Evaluation and Testing**

* **Model Evaluation**: Regularly measuring the accuracy and effectiveness of both the customer segmentation and the recommendation models against a test data set. This involves selecting and calculating suitable metrics for classification and recommendation systems.
* **Model Improvement**: Continually refining the models based on evaluation results and feedback, ensuring they remain accurate and relevant in a changing business environment.
* **Test Scenarios**: Running the models through various test scenarios to ensure their robustness. These scenarios will be designed to emulate potential real-world conditions the models may encounter.
* **Performance Reporting**: Delivering detailed performance reports of the models, including key metrics, to provide insights into their accuracy and efficiency, aiding in decision-making and future improvements.

**Project Deliverables:**

**Customer Segmentation Model**

* **Data Extraction**: Deliver a method for extracting and cleansing data from Hadoop tables (Customer Master, Account Master, Transaction Pool).
* **Feature Engineering**: Deliver a strategy for selecting, transforming, and engineering features from raw data that are suitable for machine learning models.
* **Categorical Representation**: Develop a system to categorize monthly transactions into understandable categories like 'Shopping', 'Investments' etc.
* **Model Building**: Design and deploy a Multioutput Classifier to categorize customers into segments.
* **Segmentation Output**: Create a new 'Segmentation' table in Hadoop, containing the output of the segmentation model, which categorizes customers based on various parameters.

**Service Recommendation Model**

* **Model Development**: Deliver a machine learning model that uses customer segmentation to recommend banking services to customers.
* **Recommendation Categories**: Design the model to suggest services such as 'Loan Recommendation', 'Investment Recommendation', 'Credit Card Recommendation', 'Insurance Recommendation' based on the customer's transaction classifications.
* **Recommendation Output**: A detailed table or report of recommendations for each customer, which can be used by the sales and marketing teams.
* **Model Evaluation**: Regular reports that measure the accuracy and effectiveness of the models against a test data set, using appropriate metrics for classification and recommendation systems.
* **Model Maintenance and Upgrade Plan**: A plan for maintaining and upgrading the model to cater to changing customer behaviors and market trends, including regular retraining of the model with new data.

**Benefits to the Bank**

* **Personalized Services**: The bank can offer more personalized services based on customers' behaviors and needs, leading to increased customer satisfaction.
* **Enhanced Customer Retention**: By offering targeted and relevant services, the bank can increase customer loyalty and reduce churn rate.
* **Improved Sales**: Recommendations derived from the model can drive higher sales, as they're tailored to each customer's preferences.
* **Efficient Resource Allocation**: By knowing which services are more likely to be adopted by which customer segments, the bank can allocate resources more efficiently.
* **Risk Mitigation**: The segmentation model can potentially highlight customer segments that exhibit risky behaviors, enabling the bank to take preventive measures.
* **Data-Driven Decision Making**: The models will enable the bank to make strategic decisions based on concrete data, enhancing overall operational efficiency.
* **Market Advantage**: Such sophisticated models can provide the bank with a competitive edge in a market where personalized service is increasingly important.
* **Insightful Customer Understanding**: Deep insights into customer behavior can enable the bank to predict trends and stay ahead of customer demands.
* **Targeted Marketing**: The segmentation model will enable targeted marketing, ensuring that the right products are marketed to the right customers, thereby increasing conversion rates.
* **Revenue Growth**: By effectively selling targeted and relevant services to customers, the bank can increase its revenue growth rate.