"IT FEELS UNFAIR THAT APPLYING FOR CREDIT LOWERS MY SCORE BECAUSE OF HARD INQUIRIES, EVEN WHEN I'M JUST SHOPPING FOR THE BEST DEAL. IT MAKES ME HESITANT TO APPLY FOR CREDIT AT ALL."

"THE CREDIT SCORING SYSTEM IS CONFUSING. I DON'T KNOW WHAT FACTORS ARE AFFECTING MY SCORE, AND THERE'S NO CLEAR EXPLANATION OF HOW MY SCORE IS CALCULATED."

Predictive Model for Credit Card Approval

Solving Applicant Uncertainty and Improving Institutional Efficiency

TEAM GROUP 5

Archana
Kalyani
Manisha
Subhash
Suvarna

TABLE OF CONTENTS

I. Executive Summary

The Predictive Credit Card Approval Project harnesses Big Data frameworks and cloud-based machine learning to streamline the credit card application process and mitigate associated risks. Utilizing a comprehensive dataset from Kaggle, this project develops a predictive model that offers real-time assessments on credit card approvals before formal applications are submitted. This pre-emptive evaluation is crucial as it helps applicants avoid the negative impacts of hard inquiries on their credit scores, which occur with each formal application and can be damaging if the application is likely to be rejected.

For financial institutions, the model significantly enhances operational efficiency by allowing them to focus resources on high-potential applications, thereby reducing the time and costs associated with processing likely rejections. This not only preserves a healthier client portfolio but also improves customer satisfaction by lowering rejection rates, potentially increasing customer loyalty.

In essence, the Predictive Credit Card Approval Project leverages advanced technology to refine the application process, benefiting both applicants and financial institutions by reducing unnecessary credit checks and focusing on quality applications. This innovation streamlines operations and enhances the overall user experience in the credit application sector.

II. Business Questions for this dataset

Why is predicting credit card approval important for applicants?
How is this prediction beneficial for financial institutions?
What factors in the dataset are most relevant for predicting credit card approval?
Why do financial institutions care about minimizing rejected applications?
What role does customer satisfaction play in this process?
How are demographic factors important for prediction?

III. Problem Identification

Credit card applications often require a thorough risk assessment, during which financial institutions evaluate an applicant’s creditworthiness. A significant downside of this process is that every formal credit card application typically results in a "hard inquiry" on the applicant’s credit report, potentially lowering their credit score, even if the application is rejected. For applicants who are unsure of their approval chances, this can create a stressful situation, as applying for credit can hurt their chances of future creditworthiness. Thus, individuals who might not qualify for a card risk damaging their credit score simply by trying to apply.

Financial institutions face another set of challenges in this process. Handling large volumes of applications, many of which will be rejected due to mismatches with the institution's credit criteria, is resource intensive. It consumes time, workforce, and operational costs. Additionally, frequent rejections negatively impact customer satisfaction, leading to damaged relationships with potential clients and harming the institution’s reputation. As a result, banks and lenders are looking for solutions that streamline the approval process while maintaining a strong and accurate credit risk assessment.

IV. Scenario for the Project

The main objective of this project is to address these challenges by developing a system that predicts whether an applicant is likely to get approved for a credit card without them needing to formally apply. This solution would analyze key applicant data, such as demographics, financial history, and credit behavior to provide a pre-

application approval prediction. By using this predictive model, applicants can determine whether they are likely to be approved before officially applying, protecting their credit score from the negative impact of multiple hard inquiries.

For financial institutions, the predictive model can help reduce the number of applications that are likely to be rejected early in the process, allowing them to focus their resources on high-quality applicants with a higher chance of approval. This not only improves operational efficiency but also enhances the customer experience, reducing the number of rejections and improving customer satisfaction. By filtering out high-risk applicants before they formally apply, financial institutions can reduce costs while still ensuring that they maintain accurate and reliable risk assessments.

V. Actual Problem to Solve

The key problem to solve is the need to accurately predict whether a credit card application will be approved or rejected based on available applicant data, without the need for a formal application that could negatively impact the applicant's credit score. This solution must:

Provide applicants with a pre-application approval prediction to help them decide whether to proceed with a formal application, protecting their credit score.
Improve the efficiency of the approval process for financial institutions by reducing the number of rejected applications they need to process.

Enhance customer satisfaction by reducing uncertainty, improving transparency, and lowering the risk of unnecessary credit score damage.

This solution benefits both the applicants and financial institutions by balancing the need for efficient credit decisions with improved customer experience.

VI. Business Description: Predictive Approval Solutions (PAS)

Predictive Approval Solutions (PAS) is a financial technology platform designed to solve these challenges by providing a pre-application credit card approval prediction tool. The platform integrates with financial institutions and uses advanced machine learning algorithms to analyze applicant data such as credit history, income, and employment details. By doing this, PAS offers users a reliable prediction of whether they are likely to be approved for a credit card, without the need for a formal application that could negatively impact their credit score.

PAS not only benefits consumers by reducing the stress and risk associated with credit card applications but also helps financial institutions optimize their approval process. By filtering out applications that are unlikely to be approved, PAS enables banks and credit card issuers to focus on high-quality applicants, reducing operational costs and improving approval rates.

VII. Who benefits from this Business?

1. **Applicants (Consumers):** Consumers are the primary beneficiaries of PAS. Many individuals, especially those with lower or uncertain credit scores, hesitate to apply for credit cards because they fear rejection and the impact on their credit score. PAS provides them with a pre-application prediction of approval based on a comprehensive analysis of their financial and credit data. This transparency allows them to make informed decisions about applying for credit without the worry of hurting their credit score

unnecessarily. It also enables consumers to take steps to improve their credit profile if needed before formally applying, giving them more control over their financial future.

2. **Financial Institutions (Banks and Credit Card Issuers):** PAS also provides significant advantages to financial institutions. Processing and reviewing applications that will likely be rejected is both costly and time-consuming. By using PAS, banks and credit card issuers can focus on applicants who have a higher probability of approval, reducing the burden on their application processing teams and lowering operational costs. In addition, PAS enhances customer satisfaction by reducing the number of rejections, leading to stronger relationships with potential clients and a more efficient onboarding process.

VIII. Dataset Overview (Source: Kaggle):

The dataset for this project is sourced from Kaggle, a well-known platform for data science competitions and public datasets. The dataset contains anonymized financial and credit-related records of applicants and includes a wide variety of features essential for predicting credit card approval outcomes. With over 500,000 records, it provides ample data to build and train an accurate machine learning model.

The project requires two datasets: the Application History and the Credit History. The Application History dataset is 54.5 MB in size and contains 438,558 columns across 18 rows. Similarly, the Credit History dataset is 15.4 MB, consisting of 100,000 columns and 3 rows.

Key Features of the Dataset:

1. Demographic Information:

- Age: Age of the applicant.
- Gender: Applicant's gender.
- Marital Status: Whether the applicant is single, married, or divorced.
- Number of Dependents: Provides insight into the applicant's family obligations.

2. Financial Data:

- Annual Income: Applicant's reported income, an essential factor for determining creditworthiness.
- Employment Status: Indicates employment stability, which is crucial for assessing the applicant's ability to manage credit obligations.
- Debt-to-Income Ratio: Reflects the applicant's debt load relative to income.
- Credit Utilization Rate: A measure of how much of the available credit is currently being used by the applicant.

3. Credit History:

- Credit Score: One of the most important factors, summarizing the applicant's creditworthiness.
- Length of Credit History: Indicates how long the applicant has been using credit.
- Number of Open Credit Accounts: Shows how many accounts are currently active.
- Payment History: Tracks whether the applicant has been consistent with on-time payments.

4. Application History:

- Previous Credit Applications: Records of whether the applicant has applied for credit before and whether they were approved or rejected.
- Reasons for Rejection: If rejected, this field provides insight into why the application was denied (e.g., poor credit score, insufficient income).

IX. Tools and Technologies:

1. Programming Language:
   Python: Python is the main language used for data analysis, model building, and deployment due to its rich ecosystem of libraries for machine learning and data manipulation.
2. Data Processing and Engineering:
   PySpark: Distributed computing framework for large-scale data cleaning, transformation, and feature engineering.
3. Machine Learning Model Development
   TensorFlow: A powerful machine learning framework for building, training, and evaluating the predictive models.
   MLlib: For distributed Machine learning tasks such as classification, regression, clustering and recommendation.
4. Model Deployment and Scalability
   Kubernetes Engine: For deploying the machine learning model in a containerized environment, ensuring scalability and reliability.
5. Data Visualization and Reporting
   Tableau or Power BI

X. Data Sources:

1) Application Record Dataset:

Source: Kaggle

Link: https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction?select=application_record.csv

Description: This dataset contains demographic and financial information about credit card applicants. It includes 438,557 records and 18 attributes related to the applicant's background, income, employment, and family status.

Attributes:

- ID: Unique identifier for each applicant.
- CODE_GENDER: Gender of the applicant.
- FLAG_OWN_CAR: Indicates whether the applicant owns a car (Yes/No).
- FLAG_OWN_REALTY: Indicates whether the applicant owns property.
- CNT_CHILDREN: Number of children the applicant has.
- AMT_INCOME_TOTAL: Total annual income.
- NAME_INCOME_TYPE: Income type (e.g., working, pensioner).
- NAME_EDUCATION_TYPE: Level of education.
- NAME_FAMILY_STATUS: Family status (e.g., single, married).
- NAME_HOUSING_TYPE: Type of housing (e.g., house/apartment).
- DAYS_BIRTH: Age of the applicant in days (negative values indicate the number of days from birth).
- DAYS_EMPLOYED: Number of days the applicant has been employed (negative values).
- FLAG_MOBIL: Whether the applicant has a mobile phone.
- FLAG_WORK_PHONE: Whether the applicant has a work phone.
- FLAG_PHONE: Whether the applicant has a personal phone.
- FLAG_EMAIL: Whether the applicant has an email.
- OCCUPATION_TYPE: Occupation of the applicant.

- CNT_FAM_MEMBERS: Number of family members.

2) Credit Record Dataset:

Source: Kaggle

Link: https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction?select=credit_record.csv

Description: This dataset tracks the credit history of applicants over time. It contains 1,048,575 records with 3 attributes that describe the status of the applicant's credit balance on a monthly basis.

Attributes:

- ID: Unique identifier matching applicants from the application record dataset.
- MONTHS_BALANCE: The month of record relative to the current date (0 is the most recent month, negative values indicate past months).
- STATUS: The status of the applicant's credit balance for that month (e.g., '0' for no balance, '1' for credit overdue, 'C' for closed).

XI. Acquisition:

How it was acquired: The data was downloaded directly from Kaggle in CSV format.

File types:

- Application Record: 54.5 MB, 438,557 rows × 18 columns. Size:53,070 KB
- Credit Record: 15.4 MB, 1,048,575 rows × 3 columns. Size: 15,007 KB

XII. Data Process:

Data Process Flow Diagram:

Figure. XII

1.Ingest Data Pipeline:

Azure Data Factory (ADF): The Orchestration Tool

Azure Data Factory is a cloud-based data integration service that enables you to create workflows for data movement and transformation. In this scenario, ADF is used to orchestrate the process of moving data from one location to another, specifically from an external source to an Azure Blob Storage container.

IngestDataPipeline:

The pipeline named IngestDataPipeline in ADF is a sequence of steps (activities) that define how data flows from source to destination.

This pipeline consists of two activities, CopyApplicationRecord and CopyCreditRecord, each of which is responsible for moving a specific file from the source to the destination.

Copy Activities:

Each "Copy data" activity in ADF performs an ETL (Extract, Transform, Load) operation for a single data source. In this case:

CopyApplicationRecord moves application_record.csv.

CopyCreditRecord moves credit_record.csv.

These activities define both the source location (where the data originates) and the destination location (where it should be stored). In this case, the source could be an external database or file location, and the destination is an Azure Blob Storage container.

ADF handles connectivity, data transfer, and any necessary transformations on the fly, ensuring that data is accurately moved to the target location.

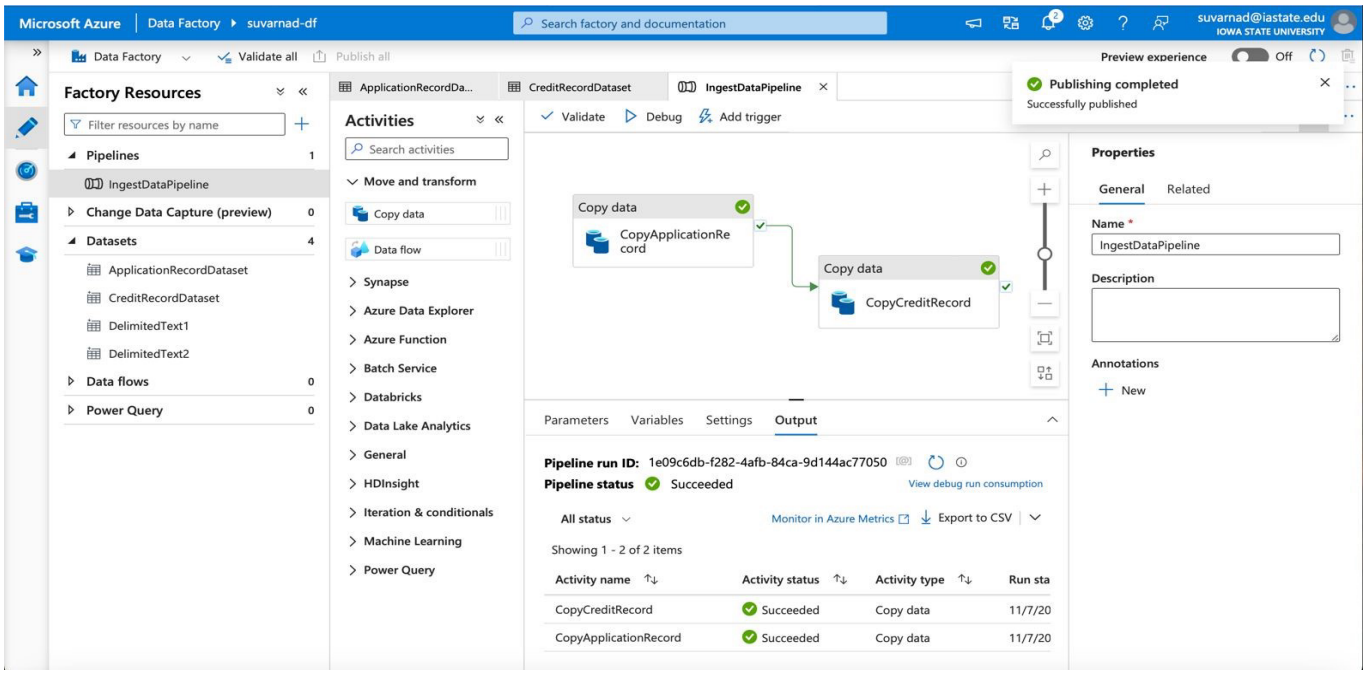

Figure. XII.1.1

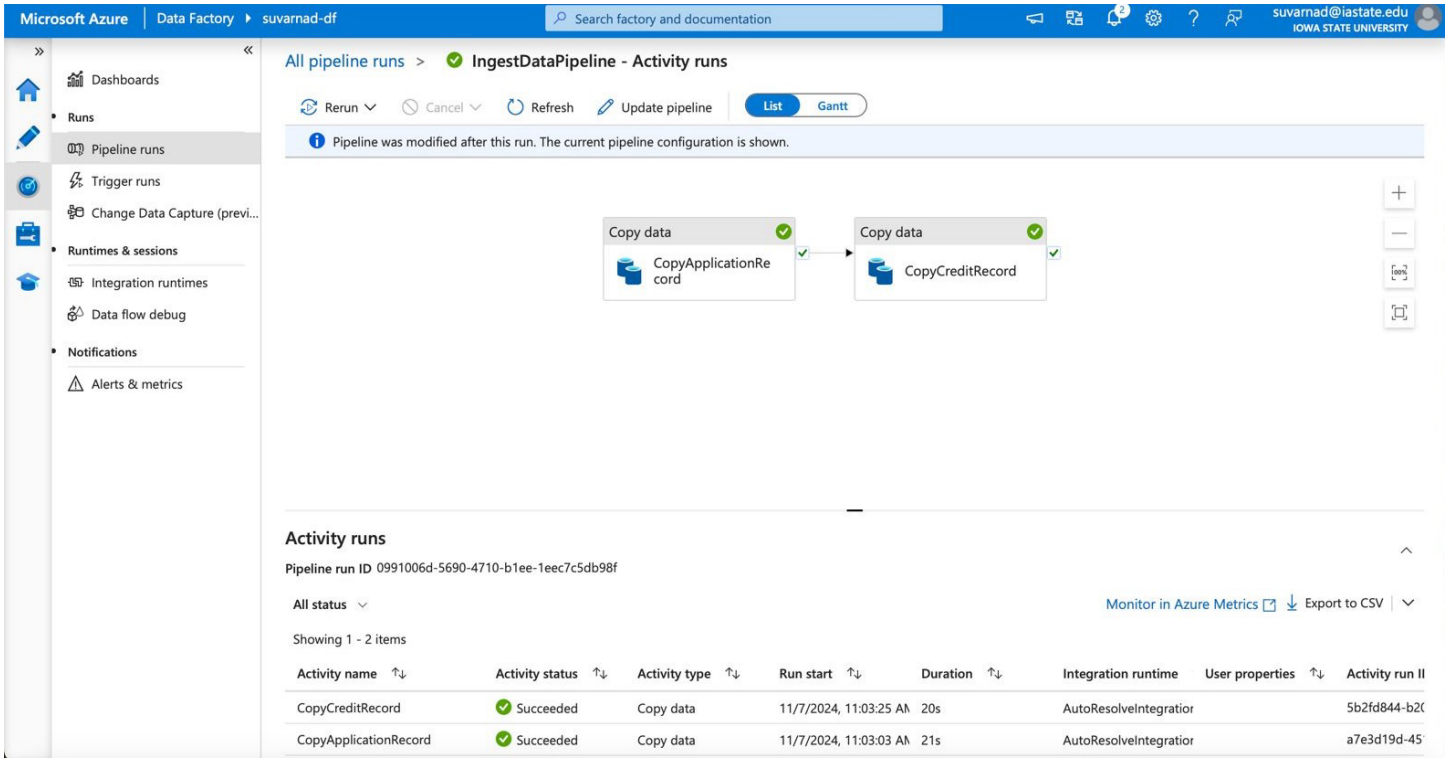

Figure. XII.1.2

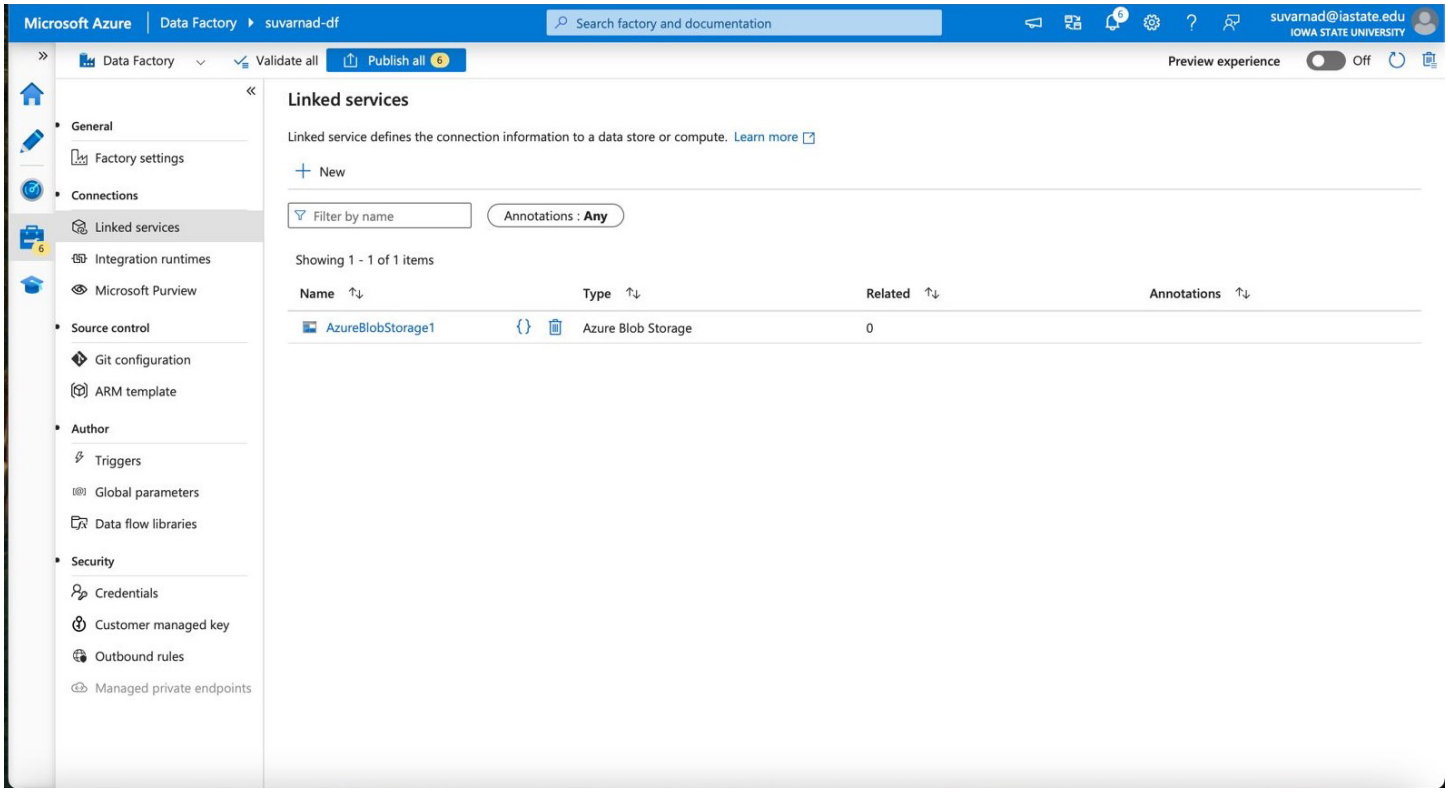

Figure. XII.1.3

2. Azure Blob Storage: The Data Storage Solution

Azure Blob Storage is a cloud-based storage solution that stores large amounts of unstructured data, such as files, documents, and media. In this setup, the Blob Storage container serves as the destination for the data ingested by ADF.

Blob Storage Container (ingested-data):

When ADF ingests data, it saves files like application_record.csv and credit_record.csv into this container.

Blob Storage supports various access tiers (Hot, Cool, Archive) to optimize storage costs based on access frequency.

Blob Types and Security:

Each file in Blob Storage is stored as a "block blob," which is optimized for large data uploads and downloads.

For security, you use SAS (Shared Access Signatures) to provide time-limited access to specific files in Blob Storage. This allows services like Synapse or Power BI to securely access data in the container without exposing storage credentials.

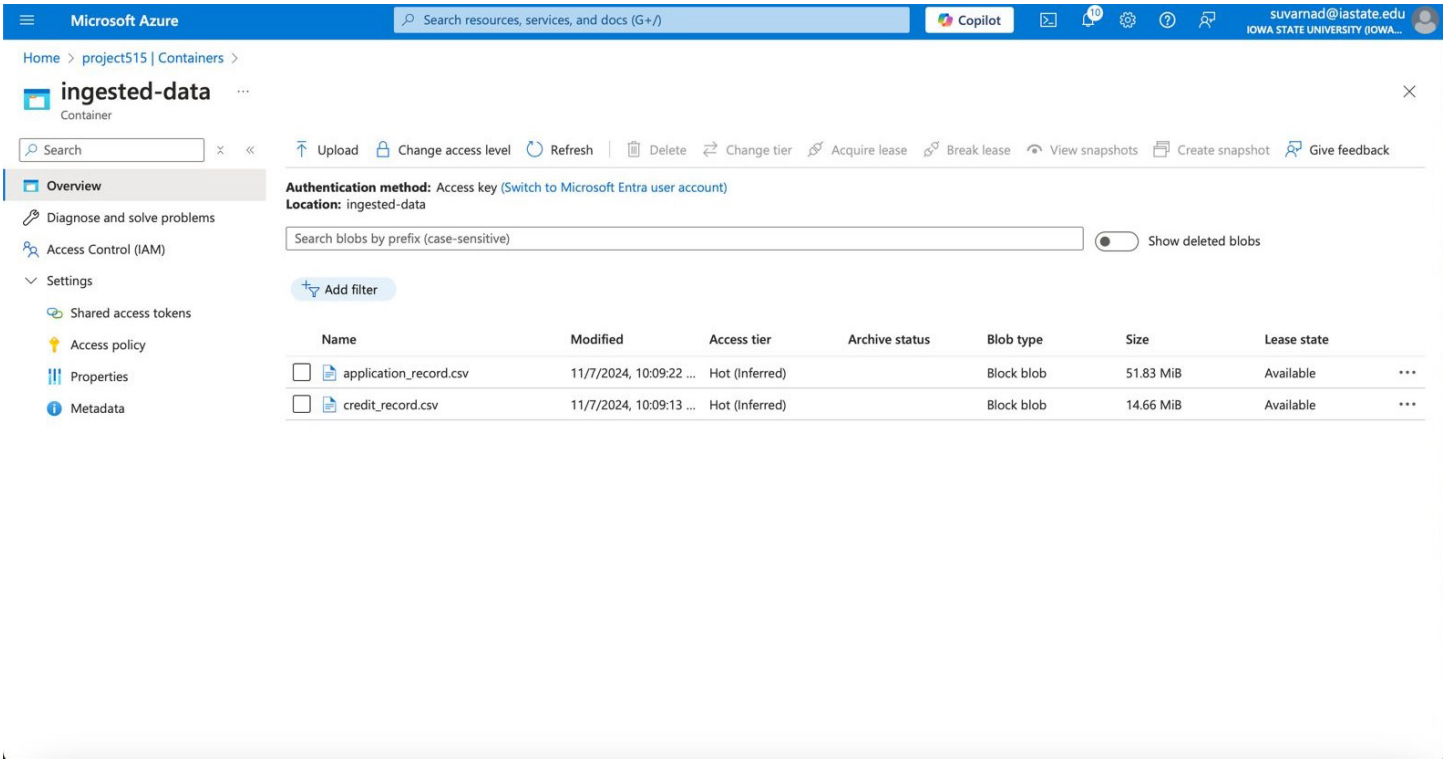

Figure. XII.2.1

3. Data Access and Processing in Azure Synapse Notebook

After data ingestion into Blob Storage, it becomes available for downstream processing and analysis. In this case, Azure Synapse is used to perform data processing on the ingested files.

Accessing Files in Synapse Notebook:

In Azure Synapse, a notebook is configured to access and process the data files stored in the ingested-data container.

Python's requests library is used to fetch data from the Blob Storage container using the SAS URLs. This allows for secure and programmatic access to the files.

Once accessed, the data is loaded into data structures like Pandas Data Frames, making it easier to perform data manipulations, analyses, and visualizations.

ADLS could serve as a destination for processed data or predictions, allowing for scalable storage and integration with other Azure analytics tools.

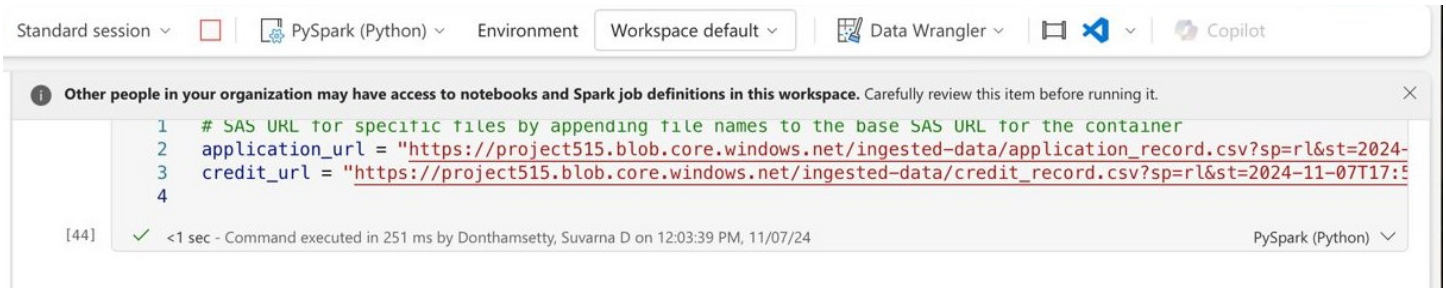

Figure. XII.3.1

Figure. XII.3.2

```
1  cosmos_endpoint = "<https://projectfall515.documents.azure.com:443/>"
2  cosmos_key = "<<B9ZBf1ewFCKlI4P9VKn0pDgkawHJA0Em8yPsM0yGCkM9jTr4omD7PfPazOtd4ODoDo3mYmusx9chACDbxCGj7g==>>"
3  database_name = "CreditRiskAnalysisDB"
4  container_name = "PredictionsContainer"
```

[107] ✓ - Command executed in 246 ms by Donthamsetty, Suvarna D on 5:58:37 PM, 11/07/24 PySpark (Python)

```
1
2  adls_url = f"abfss://<ingested-data>@<project515>.dfs.core.windows.net/predictions_output"
3  spark.conf.set(f"fs.azure.sas.<ingested-data>.<project515>.dfs.core.windows.net", "<sp=rl&st=2024-11-08T00:34:5
4
```

[114] ✓ - Command executed in 266 ms by Donthamsetty, Suvarna D on 6:36:22 PM, 11/07/24 PySpark (Python)

Figure. XII.3.3

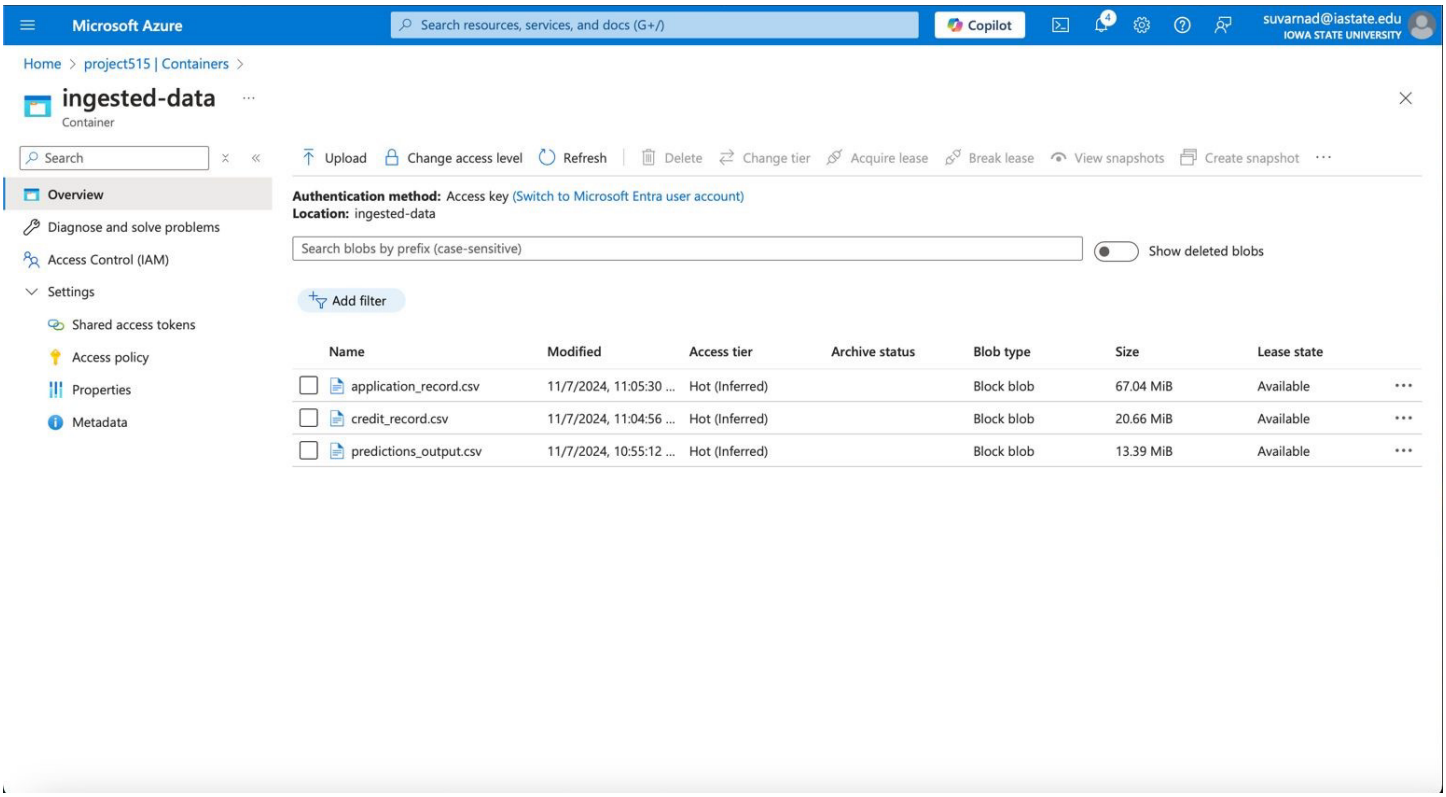

Figure. XII.3.4

4. Power BI in Fabric: Data Visualization and Analysis

After processing the data, you move to the analysis phase in Power BI. Power BI allows you to create interactive visualizations, dashboards, and reports from the data stored in Azure.

Connecting to Blob Storage in Power BI:

Power BI in Fabric is configured to connect to the Blob Storage container where the processed data is stored.

By specifying the URL of predictions_output.csv (the processed file), Power BI can access this data directly from Blob Storage.

The connection settings in Power BI use anonymous authentication, given that the SAS URL already restricts access based on permissions and expiration time.

Data Preview and Import:

Power BI provides a preview of the data in predictions_output.csv, allowing you to verify the structure and contents of the dataset before loading it for visualization. After confirming the data structure, you import it into Power BI's workspace, making the fields available in the data pane.

Building Visualizations:

With the data loaded, you can create visualizations to analyze the dataset. For example, you can drag fields like AMT_INCOME_TOTAL, DAYS_BIRTH, and CNT_CHILDREN onto the report canvas to explore relationships and trends.

Power BI's visualization tools allow you to create charts, tables, and interactive dashboards to uncover insights and patterns in the data.

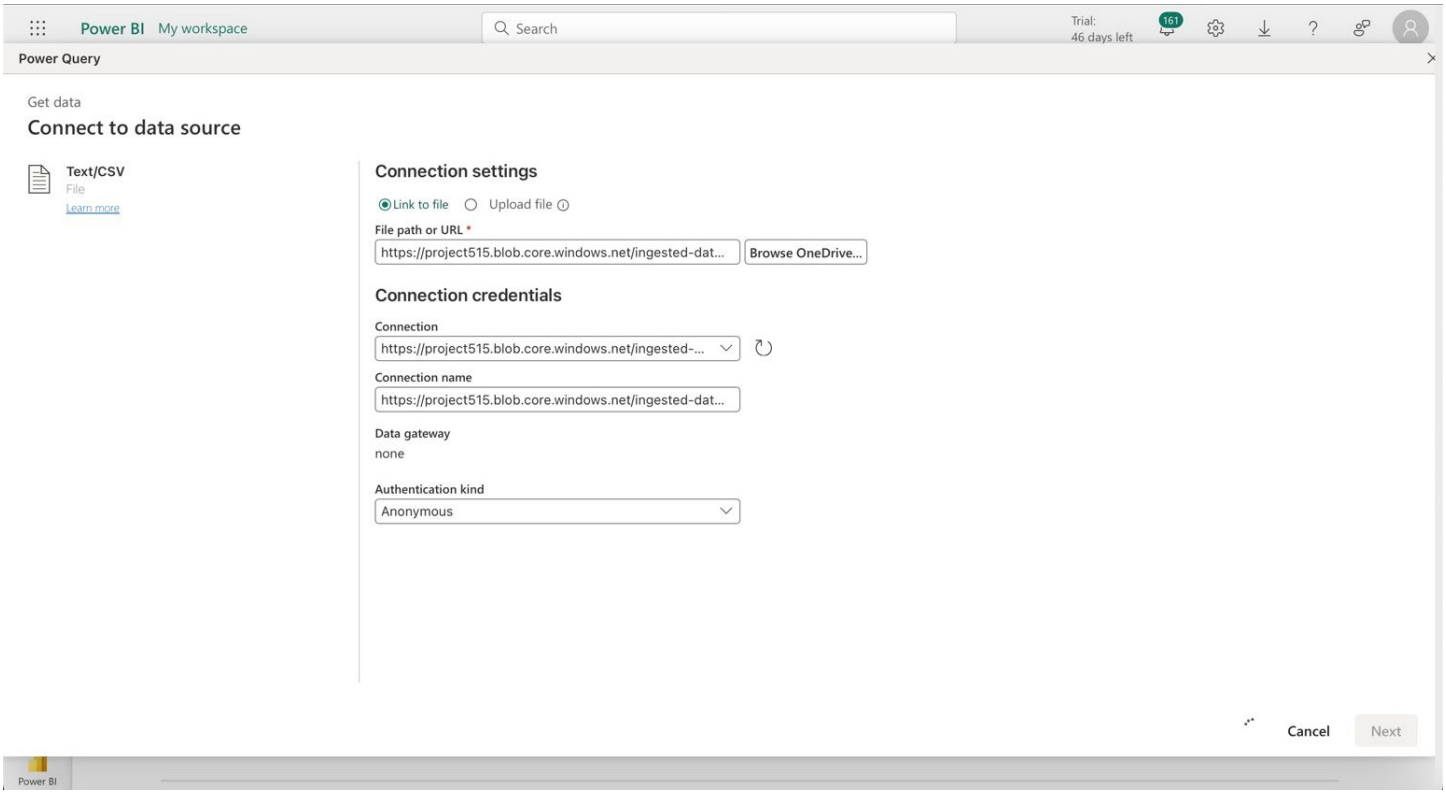

Figure. XII.4.1

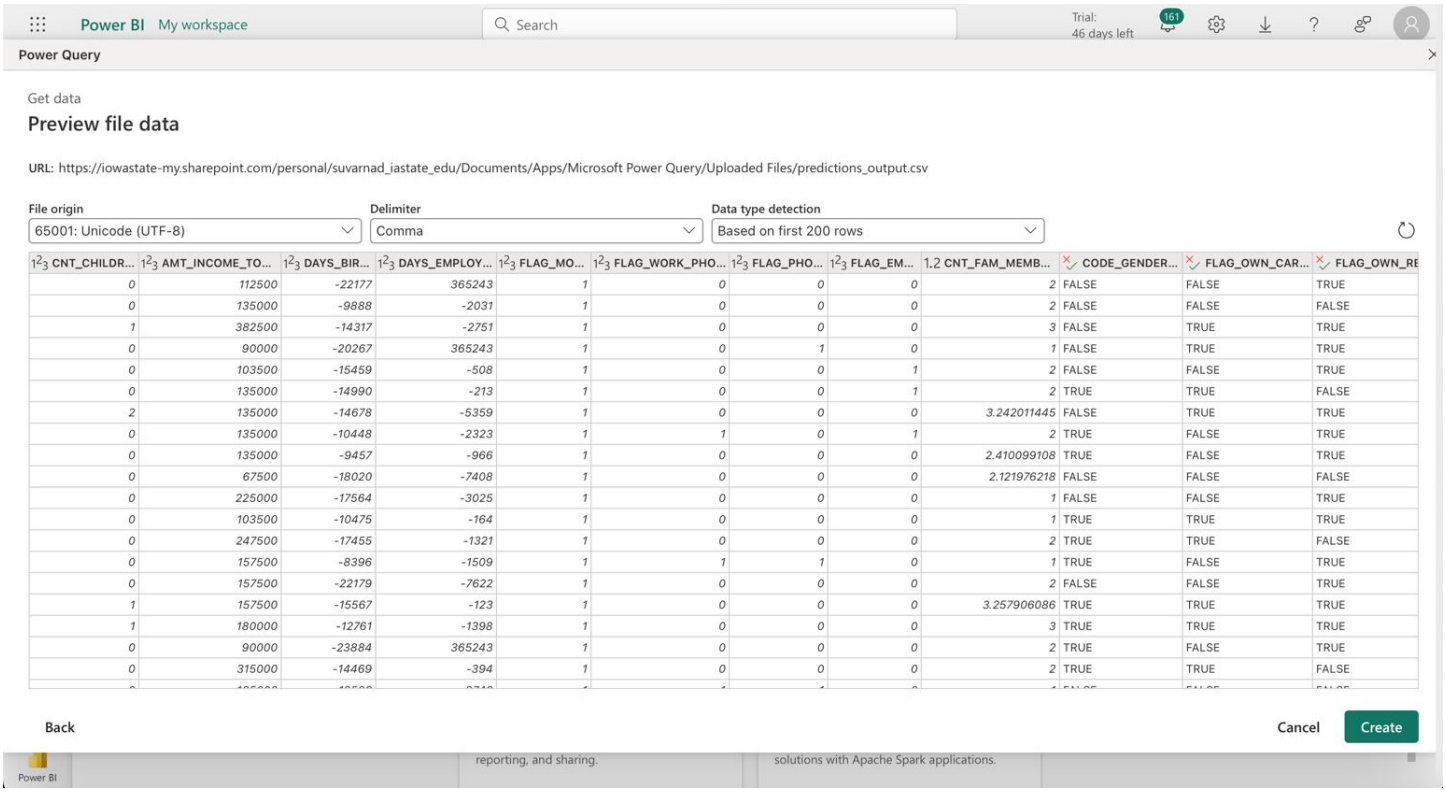

Figure. XII.4.2

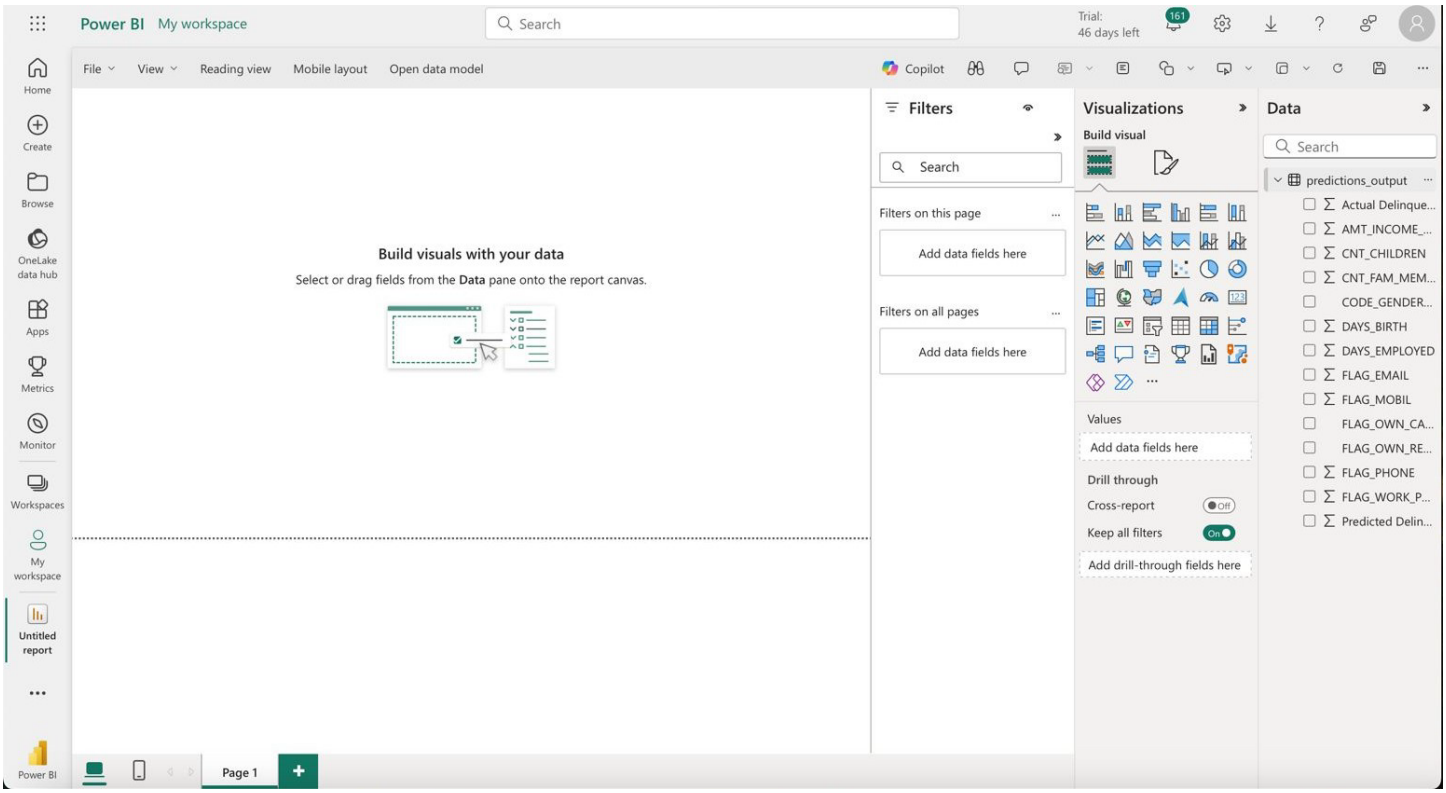

Figure. XII.4.3

End-to-End Credit Risk Prediction Pipeline Using PySpark and Azure Data Lake Integration:

Data Loading and Reading:

- The code fetches two datasets from Azure Blob Storage: application_record.csv and credit_record.csv.
- The files are read into pandas Data Frames, displayed to confirm the data, and then converted to PySpark Data Frames for further processing.

Data Merging:

- The application_data and credit_data Data Frames are merged on the ID column.
- A new column, Delinquency, is created based on the STATUS values in credit_data, where values from "0" to "5" indicate delinquency (assigned as 1), and others are marked as non-delinquent (0).
- An aggregated column Delinquency is created by grouping on ID, taking the maximum delinquency status, and then joined with application_data to ensure each application has a delinquency label.

Data Transformation:

- Dropping Unnecessary Columns: Columns such as ID, NAME_INCOME_TYPE, NAME_EDUCATION_TYPE, NAME_FAMILY_STATUS, NAME_HOUSING_TYPE, and OCCUPATION_TYPE are removed as they are not required for model training.
- Handling Missing Values: Missing values in numeric columns are filled with zeroes.
- Encoding Categorical Features: StringIndexer is applied to categorical columns to convert string labels to numerical indices, followed by OneHotEncoder for one-hot encoding, creating binary columns for categorical values.

Feature Engineering:

- A Vector Assembler is used to combine all feature columns into a single features column, which will be used as the input for the machine learning model.

Data Splitting:

- The data is split into training and testing sets with a 70-30 ratio.

Balancing the Data:

- An oversampling technique is applied to balance the training dataset, where the minority class (delinquent cases) is oversampled to match the majority class (non-delinquent cases).

Model Training:

- A RandomForestClassifier is configured with parameters for the number of trees, max depth, and seed for reproducibility.
- The model is trained on the balanced training dataset.

Prediction and Evaluation:

- Predictions are generated on the test dataset.
- A Binary ClassificationEvaluator calculates the Area Under the ROC curve (AUC) as the primary evaluation metric.
- Additionally, a classification report is attempted to be generated using scikit-learn's classification_report.

Saving Predictions:

- Finally, the configuration for saving output data to Azure Data Lake Storage (ADLS) is set up to save predictions.

Data Flow Summary:

- Data was ingested from Azure Blob Storage into Power BI in Fabric through Azure Data Factory.
- Data cleaning, transformation, and preparation were done in Power Query, and the final dataset was visualized in Power BI.

Software and Systems Used

- **Apache Spark**:

  This distributed computing framework was employed to efficiently handle large-scale data processing tasks. Spark's ability to process data in parallel across a cluster made it suitable for analyzing substantial datasets quickly and effectively. The PySpark interface provided flexibility to write data processing scripts in Python, facilitating a smooth workflow.

- **Python**:

  Python served as the primary programming language for the data processing workflow. Its rich ecosystem of libraries and ease of use made it an ideal choice for implementing data transformations, feature engineering, and machine learning tasks.

- **Scikit-learn**:

  This popular machine learning library in Python was utilized to evaluate model performance. It provided various metrics for assessing the effectiveness of the classification model, including classification reports, which summarize precision, recall, and F1-score, helping to gauge model accuracy.

- **CSV Files**:

  Data was read from and written to CSV format, a widely used file type for datasets due to its simplicity and ease of management. The choice of CSV allowed for straightforward data handling, making it easy to import and export data between different systems.

- **Power BI**:

  After generating output in CSV format, Power BI was used for data visualization. This business analytics tool enabled the creation of interactive reports and dashboards, facilitating the exploration of insights

derived from the model's predictions and various data attributes. Power BI's user-friendly interface allowed for effective communication of findings to stakeholders.

Steps Taken

- **Session Initialization:**

  A Spark session was created to enable Spark functionalities. This is a prerequisite for any data processing task in Spark.

- **Data Loading:**

  Two datasets were loaded from CSV files:

  - One containing customer application data.
  - The other detailing credit histories.

- **Data Merging:**

  The two datasets were combined into a single Data Frame using a join operation based on a common identifier (customer ID). This merged dataset contained all relevant information for each customer.

- **Delinquency Calculation:**

  A new column was introduced to indicate whether a customer was delinquent. This was determined by analyzing the status of each customer's credit records, converting various statuses into a binary delinquency indicator (1 for delinquent, 0 otherwise).

- **Data Aggregation:**

  The dataset was grouped by customer ID to aggregate the delinquency status. This ensured that if a customer had any delinquency status in their records, it was reflected in the final dataset.

- **Data Filling:**

  After aggregation, any missing values in the delinquency column were filled with zero, indicating no delinquency for those customers.

- **Feature Engineering:**

  Categorical columns were identified and prepared for modeling. This involved converting them into numerical formats through indexing and encoding, making them suitable for machine learning algorithms.

- **Feature Assembly:**

  All features were combined into a single vector that could be used as input for the model. This vector included both the original numerical features and the newly encoded categorical variables.

- **Data Splitting:**

  The final dataset was divided into training and testing subsets. This split is crucial for evaluating the model's performance on unseen data.

- **Data Balancing:**

  To address class imbalance in the dataset (more non-delinquent than delinquent records), the minority class was oversampled. This ensures that the model can learn effectively from both classes during training.

- **Model Training:**

  A Random Forest classifier was trained on the balanced training dataset. This model is effective for binary classification tasks like predicting delinquency.

- **Predictions:**

  The trained model was used to make predictions on the test dataset. This step assesses how well the model generalizes to new data.

- **Results Evaluation:**

  The predictions were compared to actual delinquency values to assess model performance. A classification report was generated to summarize the results, highlighting metrics such as precision, recall, and F1-score.

- **Model Accuracy (AUC):**

  The Random Forest model achieved an AUC (Area Under the ROC Curve) accuracy of approximately 0.861 (86.1%).

- **Classification Report:**

Precision:

- For class 0 (non-delinquent): 0.84
- For class 1 (delinquent): 0.88

Recall:

- For class 0: 0.89
- For class 1: 0.83

Score:

- For class 0: 0.86
- For class 1: 0.86

Overall Metrics:

- Accuracy: 0.86 (86%)
- Macro average (average of precision, recall, and F1 across both classes): 0.86
- Weighted average (weighted by support for each class): 0.86

```
1  evaluator = BinaryClassificationEvaluator(labelCol="Delinquency", rawPredictionCol="rawPrediction", metricN
2  accuracy_rf = evaluator.evaluate(predictions)
3  print("Random Forest Model Accuracy (AUC):", accuracy_rf)
```

✓ - Command executed in 11 sec 994 ms by Donthamsetty, Suvarna D on 5:23:33 PM, 11/07/24 PySpark (Python)

```
Random Forest Model Accuracy (AUC): 0.8614004665215449
```

Figure. XII.4.4

```
1  try:
2      y_test = test_data.select("Delinquency").toPandas()
3      y_pred_rf = predictions.select("prediction").toPandas()
4      print("Classification Report:\n", classification_report(y_test, y_pred_rf))
5  except Exception as e:
6      print("Error during conversion or classification report generation:", e)
```

[73] ✓ - Command executed in 19 sec 138 ms by Donthamsetty, Suvarna D on 5:24:34 PM, 11/07/24 PySpark (Python)

```
Classification Report:
                precision    recall  f1-score   support

           0       0.84      0.89      0.86    121904
           1       0.88      0.83      0.86    122029

    accuracy                           0.86    243933
   macro avg       0.86      0.86      0.86    243933
weighted avg       0.86      0.86      0.86    243933
```

Figure. XII.4.5

- **Data Export:**

  Finally, the predictions, along with the actual values, were saved to a CSV file for further analysis or reporting.

New Columns Created

- **Predicted Delinquency:**

  A binary indicator column that represents whether a customer has ever been delinquent based on their credit status.

Data Filtering

Filtering was applied to determine delinquent customers based on their credit status, and also when separating majority and minority classes for balancing.

Data Aggregation

The dataset was aggregated to ensure that the maximum delinquency status for each customer was captured, consolidating multiple records into a single representative record.We generated and exported to a CSV file, ready for further analysis or reporting.

XIII. Assumptions Made

Data Collection Method

- **Random Sampling:**

  It was assumed that the data used for the analysis was collected in a manner from kaggle that approximated random sampling. This means that the dataset is expected to be representative of the overall population, which is crucial for generalizing findings.

- **Bias in Collection:**

  There is an inherent assumption that the data collection methods did not introduce significant bias. However, it is acknowledged that if certain demographic groups were underrepresented or overrepresented, the results could be skewed.

Completeness of Data

- **Missing Data**:

  It was assumed that missing values in the dataset were either minimal or handled appropriately. While missing data was filled for the delinquency column, there could still be unaddressed gaps in other features that might affect the analysis.

- **Accurate Reporting:**

  It was assumed that the data provided in the application and credit records is accurate and truthful. Any inaccuracies in self-reported data could affect the validity of the analysis.

Independence of Observations

It was assumed that individual observations (customers) in the dataset are independent of one another. This means that the behavior of one customer does not influence another, which is an essential condition for many statistical analyses.

Variance Assumptions

Homogeneity of Variance: It was assumed that the variance in delinquency rates across different demographic groups was relatively constant. If variances are significantly different, it could impact the effectiveness of the classification model.

Sufficient Sample Size

It was assumed that the sample size was large enough which is 438,557 records and 18 attributes to provide reliable results. A larger sample size generally leads to more robust statistical inferences and better model performance. The split between training and testing datasets was also assumed to provide adequate data for training without overfitting.

Normal Distribution

While not strictly necessary for all machine learning models, it was assumed that certain features may follow a normal distribution. This assumption is relevant for statistical analyses and may impact the interpretation of results.

Feature Relevance

It was assumed that the selected features both numerical and categorical are relevant for predicting delinquency. While exploratory data analysis may have been conducted, there may still be unconsidered features that could influence the predictions.

Predictor Independence

It was assumed that the predictors used in the model do not exhibit significant multicollinearity. High correlation among predictors can lead to unreliable estimates and reduced interpretability of the model.

Model Performance

It was assumed that the Random Forest classifier would effectively capture the relationships in the data and generalize well to unseen data. However, the performance of the model relies heavily on the underlying assumptions about data distribution and feature relevance.

XIV. Visualizations

1. Clustered Bar Chart: Customers by Education Type

This visualization provides a comparative analysis of customer distribution across different education types, including Higher Education, Secondary/Secondary Special, Incomplete Higher, and Lower Secondary. The chart highlights that the Secondary/Secondary Special education category has the highest count of customers at 173,000, while the Incomplete Higher education category has the lowest count at just 3,000. This disparity suggests that the educational attainment level significantly influences customer demographics, indicating a potential target group for outreach or marketing efforts.

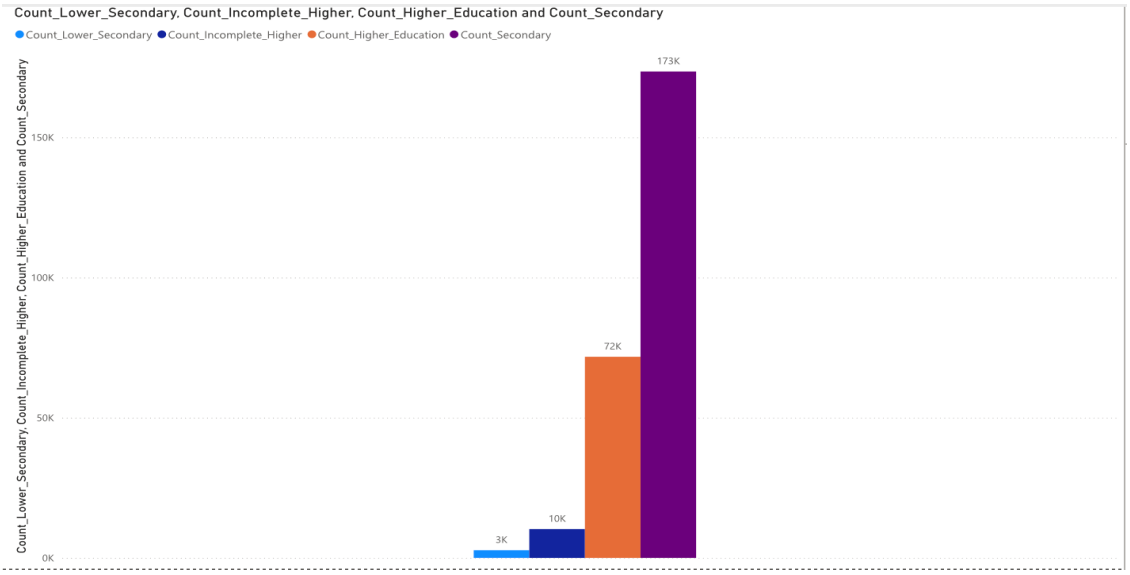

Figure. XIV.1

2. Clustered Column Chart: Average Income by Education Type

The average income across various education types is depicted in this clustered column chart. The analysis reveals that individuals with Secondary/Secondary Special education have the highest average income, while those with Incomplete Higher education exhibit the lowest average income. This correlation implies that higher educational qualifications may lead to better income opportunities, providing insights into the socioeconomic factors influencing financial stability among customers.

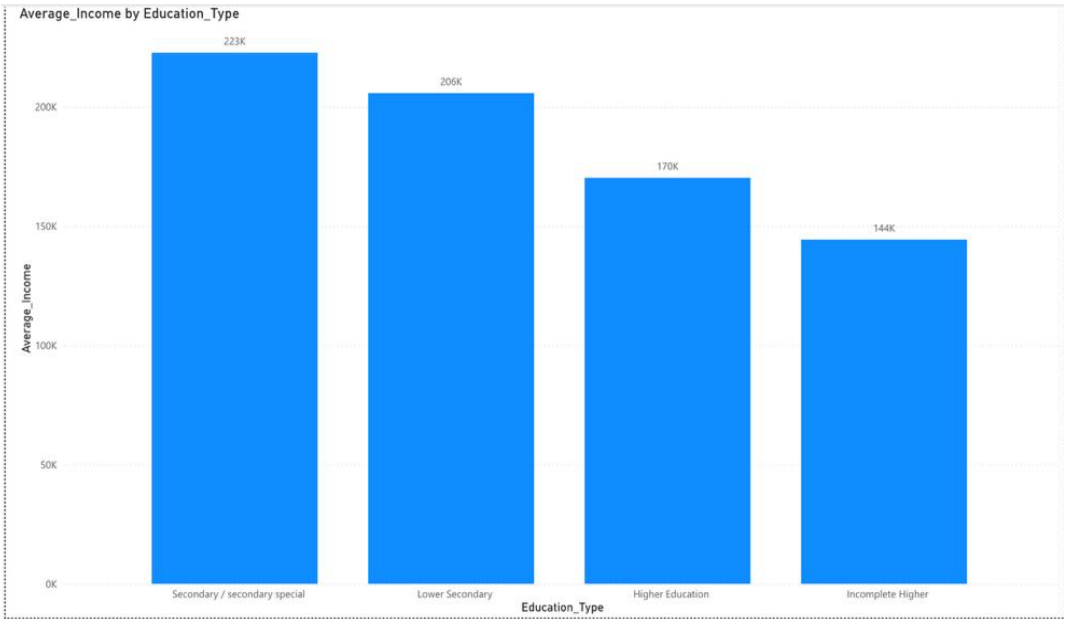

Figure. XIV.2

3. Clustered Column Chart: Approval Rate by Education Type

This chart illustrates the approval rates for credit among different education types. Notably, the Secondary/Secondary Special category has the highest approval rate at 0.48%. This information can be crucial for understanding the relationship between educational background and creditworthiness, potentially guiding lending strategies.

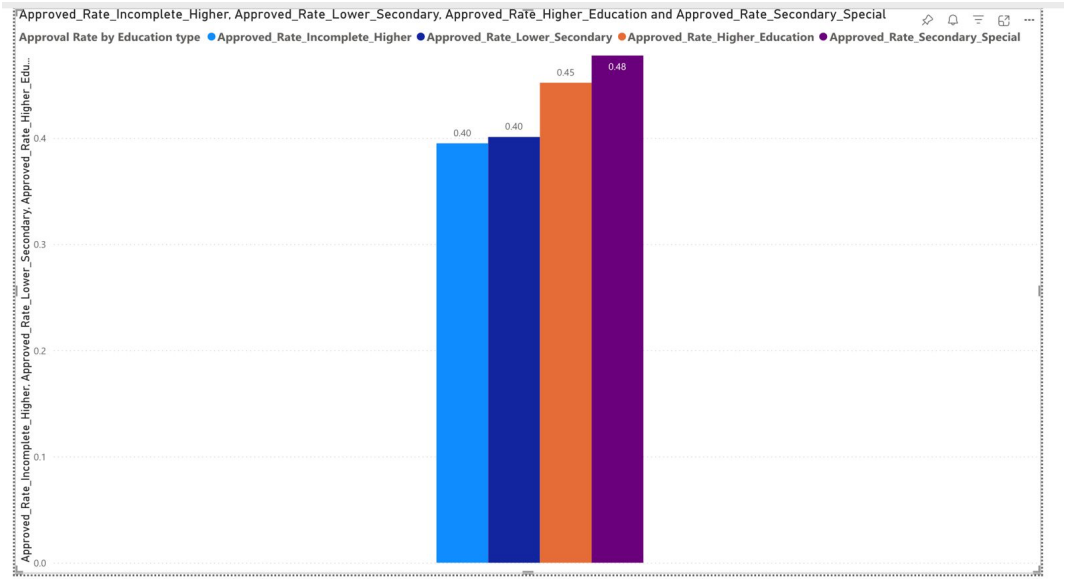

Figure. XIV.3

4. Clustered Column Chart: Count of Income Types

The tree map visualizes the distribution of customers across different income types: Pensioner, State Servant, and Working. The analysis indicates a significant prevalence of the Working income type, suggesting that most customers are actively employed. This insight can help tailor products and services to meet the needs of this demographic.

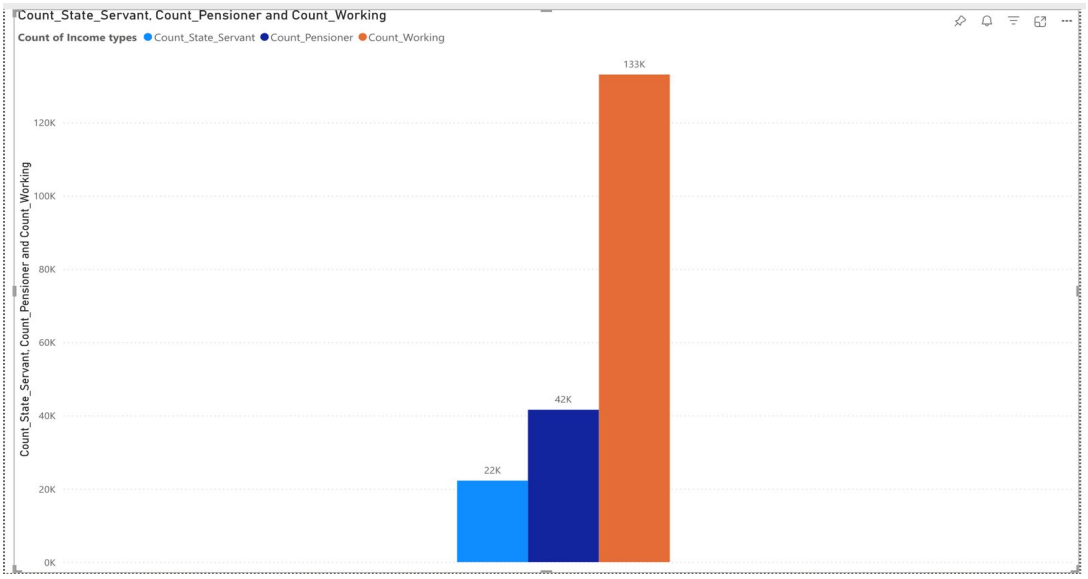

Figure. XIV.4

5. Clustered Column Chart: Average Income by Income Type

This clustered bar chart presents the average income for each income type, revealing that State Servants have the highest average income at 200,000. This finding underscores the financial advantages associated with certain employment categories and can inform decisions related to product offerings and marketing strategies targeting high-income individuals.

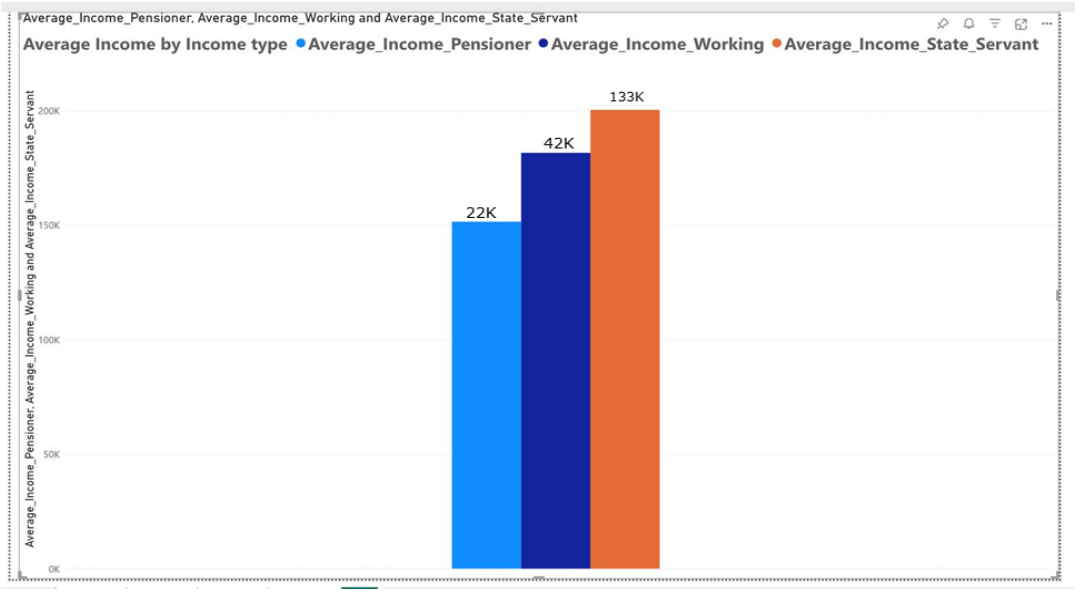

Figure. XIV.5

6. Clustered Bar Chart: Approved Count by Income Bracket

In this visualization, customers are categorized into income brackets: High Income (≥150,000), Medium Income (50,000 - 150,000), and Low Income (<50,000). The chart shows a significantly higher approved count for high-income individuals at 71,000, compared to just 1,000 approvals for low-income customers. This disparity highlights the correlation between income levels and approval likelihood, emphasizing the need for targeted financial products for different income groups.

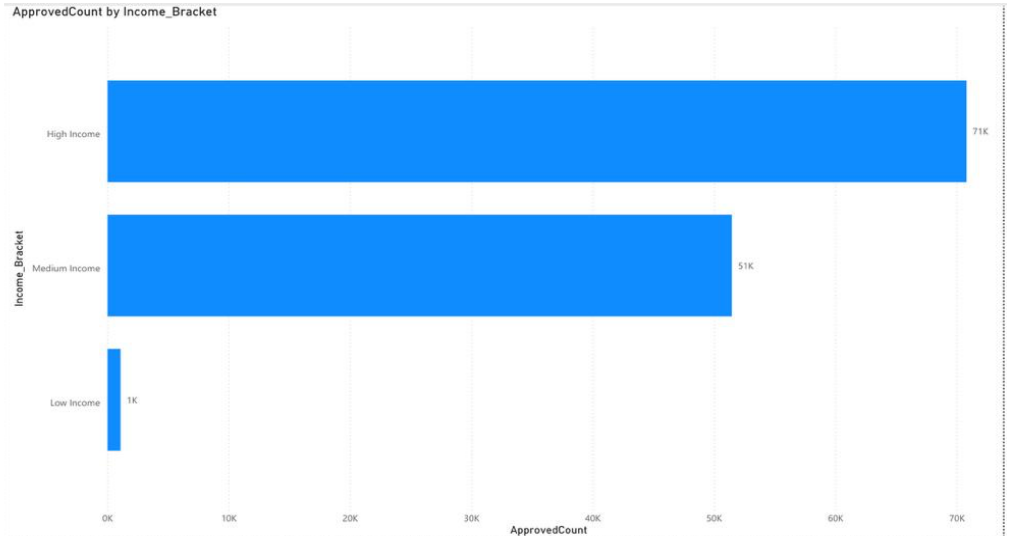

Figure. XIV.6

7. Clustered Column Chart: Count of Customers by Housing Type

This visualization displays the number of customers residing in various housing types, including House/Apartment, Rented Apartment, Co-op Apartment, Municipal Apartment, Office Apartment, and Living with Parents. The data reveals that the House/Apartment category has the highest customer count at 222,000, indicating a preference or trend toward owning or renting traditional living spaces.

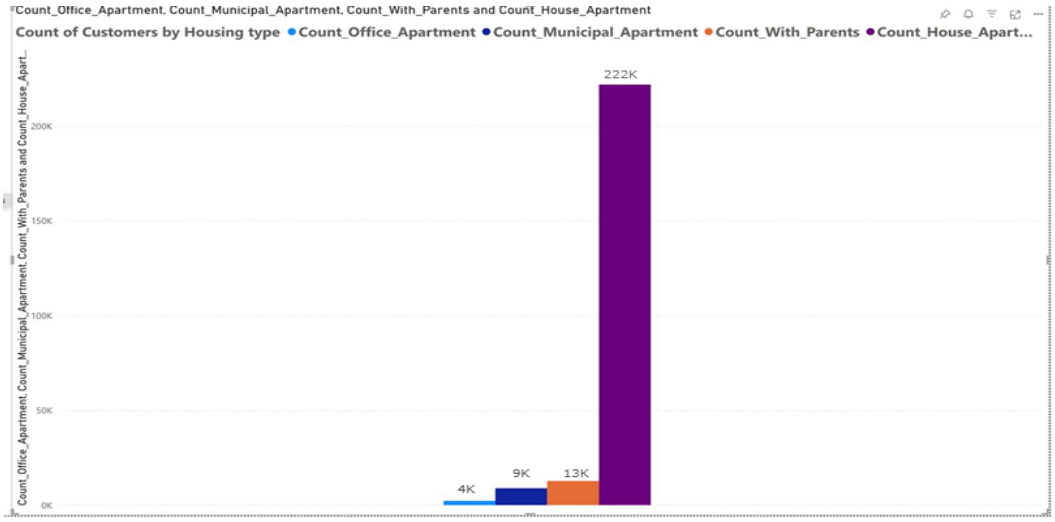

Figure. XIV.7

8. Clustered Bar Chart: Approved Count by Housing Type

The approved count for credit applications is analyzed across different housing types in this clustered bar chart. The House/Apartment category again stands out with the highest approved count of 113k, reinforcing the idea that home ownership or stable housing may correlate with credit approval success.

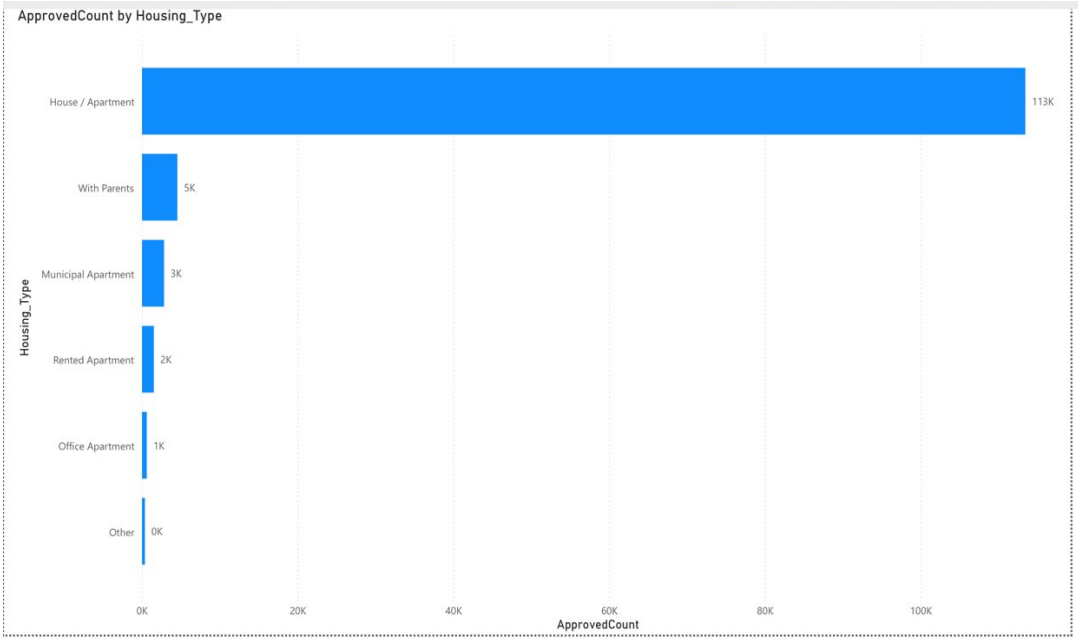

Figure. XIV.8

9. Clustered Column Chart: Apartment Type for High-Income Customers

This visualization compares the types of apartments owned by high-income customers. The results show that a substantial number of high-income individuals reside in House/Apartment types (129,000), while Office Apartments are the least common, with only 2,000 occupants. This trend suggests that higher-income customers prefer more traditional housing options over less conventional ones.

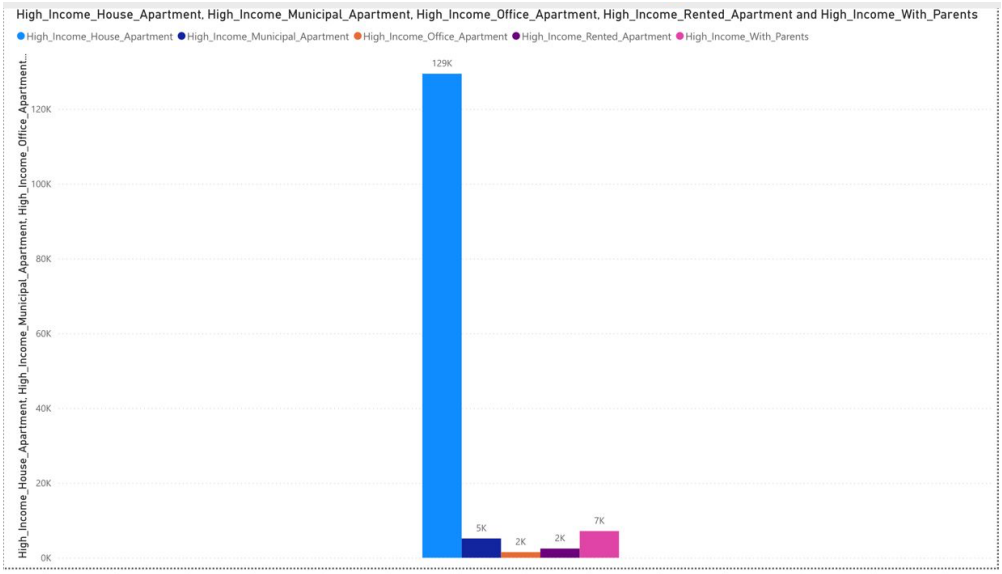

Figure. XIV.9

10. Clustered Bar Chart: Approved Count by Car Ownership

The relationship between car ownership and credit approval is explored in this clustered bar chart. The data indicates that individuals who own cars have a significantly higher number of approved credit applications. This visualization can inform lending practices, suggesting that car ownership may be a factor in assessing creditworthiness.

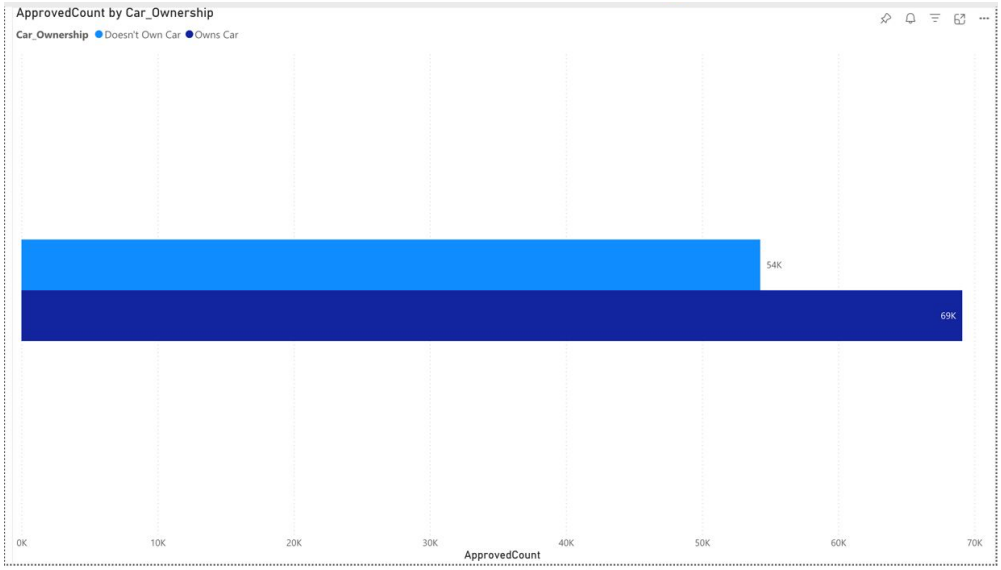

Figure. XIV.10

11. Matrix: Count of Children by Customers

This matrix compares the number of children among customers, revealing that the highest count is for customers with no children (172,759), followed by those with one child (48,288). This insight into family structure can be useful for understanding customer demographics and tailoring services that consider family-related financial needs.

Children_Range	Count_of_Children_Range
Total	**243,933.00**
1 Child	48,288.00
2 Children	19,994.00
3-4 Children	2,771.00
5+ Children	121.00
No Children	172,759.00

Figure. XIV.11

12. Clustered Column Chart: Approval Rate for Customers with Children

This visualization focuses on the approval rate for customers with children. The chart indicates that out of 71,000 individuals who applied for credit, 34,000 received approval. This information suggests that having children may influence credit approval rates, highlighting a potential area for targeted marketing or support services.

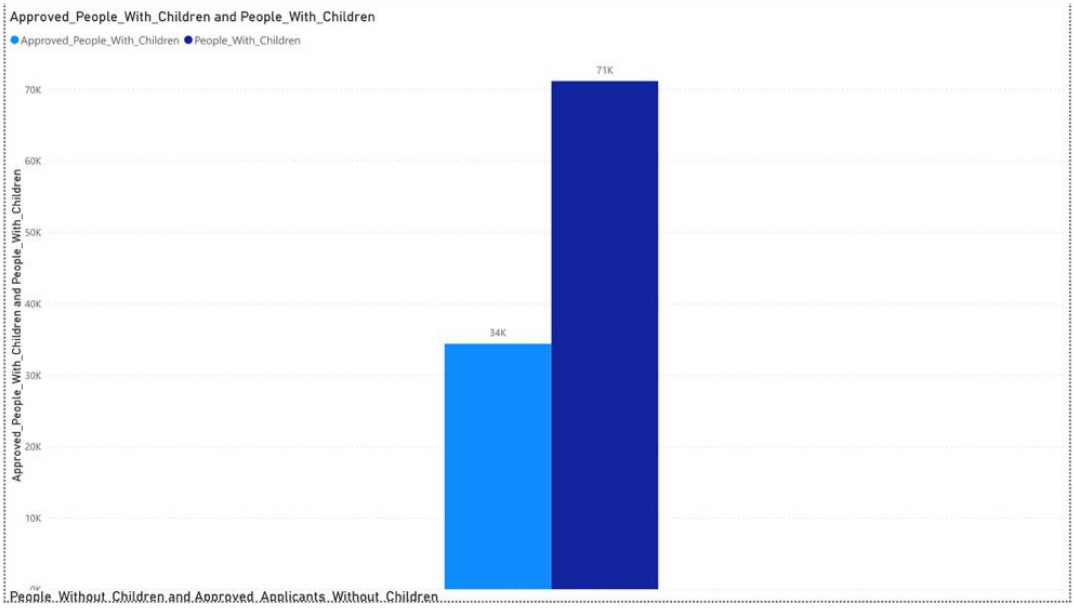

Figure. XIII.12

13. Clustered Column Chart: Approval Rate for Customers without Children

This visualization focuses on the approval rate for customers with children. The chart indicates that out of 173,000 individuals who applied for credit, 122,000 received approval, resulting in an approval rate of approximately 70.6%. This information suggests that having children may influence credit approval rates, highlighting a potential area for targeted marketing or support services. The relatively high approval rate for this demographic could inform strategies for developing products that cater specifically to families, such as childcare support services or family-oriented financial planning resources.

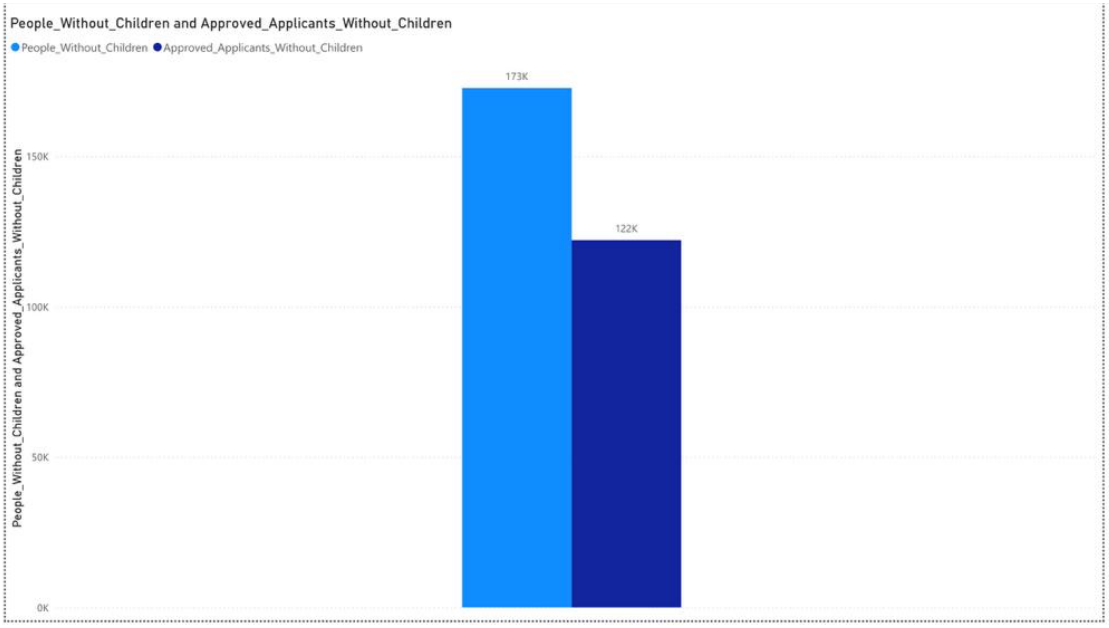

Figure. XIV.13

14. Clustered Bar Chart: Approved Count by Gender (Code_Gender_M)

This visualization focuses on the number of credit approvals based on gender, where male is represented by 1 and female by 0. The chart shows that out of the total applicants, 76,000 female applicants were approved for credit, while 47,000 male applicants received approval. This indicates that, in this dataset, the approval rate for females is higher than for males. The analysis could suggest that gender might play a role in credit approval patterns, although further investigation would be necessary to understand if this is due to other factors like income levels, credit history, or loan amount. Such insights could be used to tailor financial products or services, ensuring they are inclusive and meet the needs of all customer segments.

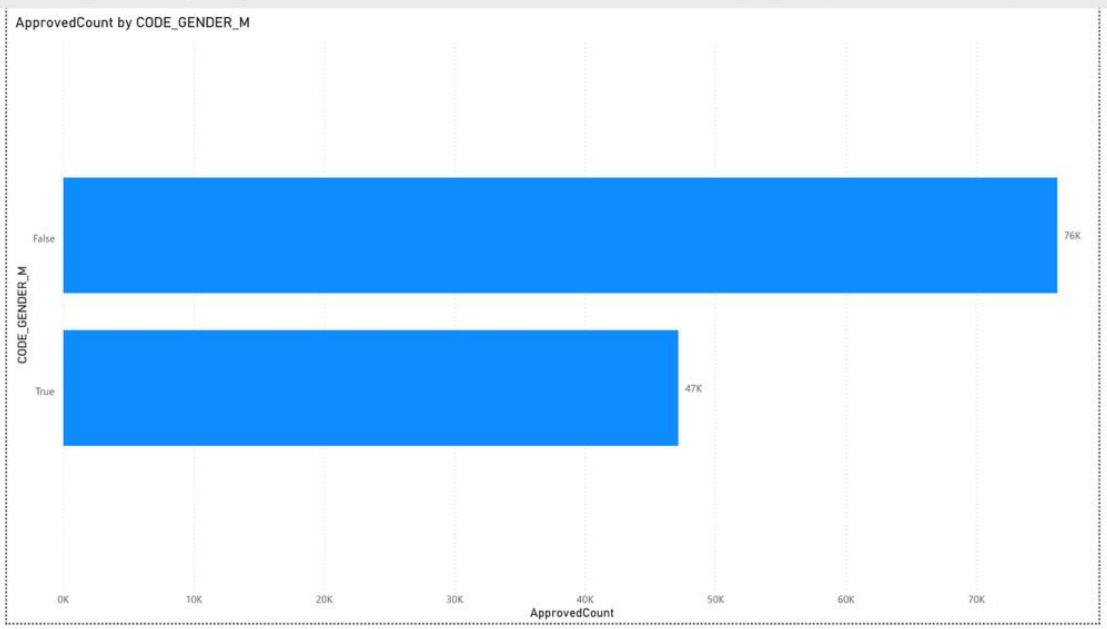

Figure. XIII.14

XV. Recommendations

The predictive model for credit card approvals provides critical insights and actionable recommendations for improving credit approval processes and enhancing customer engagement. The analysis highlights that high-income individuals have notably higher credit approval rates, emphasizing the importance of income in lending decisions. Families with children and individuals with stable housing, such as owning or renting houses/apartments, emerge as key demographic segments, indicating the potential for targeted financial products. Educational attainment also plays a significant role, with Secondary/Secondary Special education holders being the largest group and showing strong approval trends. Additionally, car ownership appears to positively correlate with credit approval rates, underscoring the influence of asset ownership on creditworthiness. Gender analysis further reveals higher approval rates among female applicants, pointing to opportunities for more inclusive financial strategies.

To leverage these findings, financial institutions should adopt the pre-application approval tool developed in this project, streamlining decision-making, reducing operational costs, and enhancing customer satisfaction. Targeted strategies, such as premium credit products for high-income groups and family-oriented financial services, can meet the diverse needs of different segments. Institutions can also design specialized offers for asset owners, focusing on car ownership as an indicator of creditworthiness. Campaigns tailored for Secondary/Secondary Special education holders and other prominent demographics can further drive engagement. Ensuring gender inclusivity while maintaining unbiased evaluation processes will foster trust and equity.

Empowering consumers through educational campaigns that demystify credit scoring and approval criteria will enable them to take proactive steps to improve their credit profiles. Continuously updating the predictive model with new data will ensure its accuracy and relevance in dynamic market conditions. By implementing these recommendations, financial institutions can enhance operational efficiency, increase customer satisfaction, and provide fairer, data-driven credit opportunities, ultimately creating a more inclusive and effective credit approval ecosystem.

XVI. Appendix

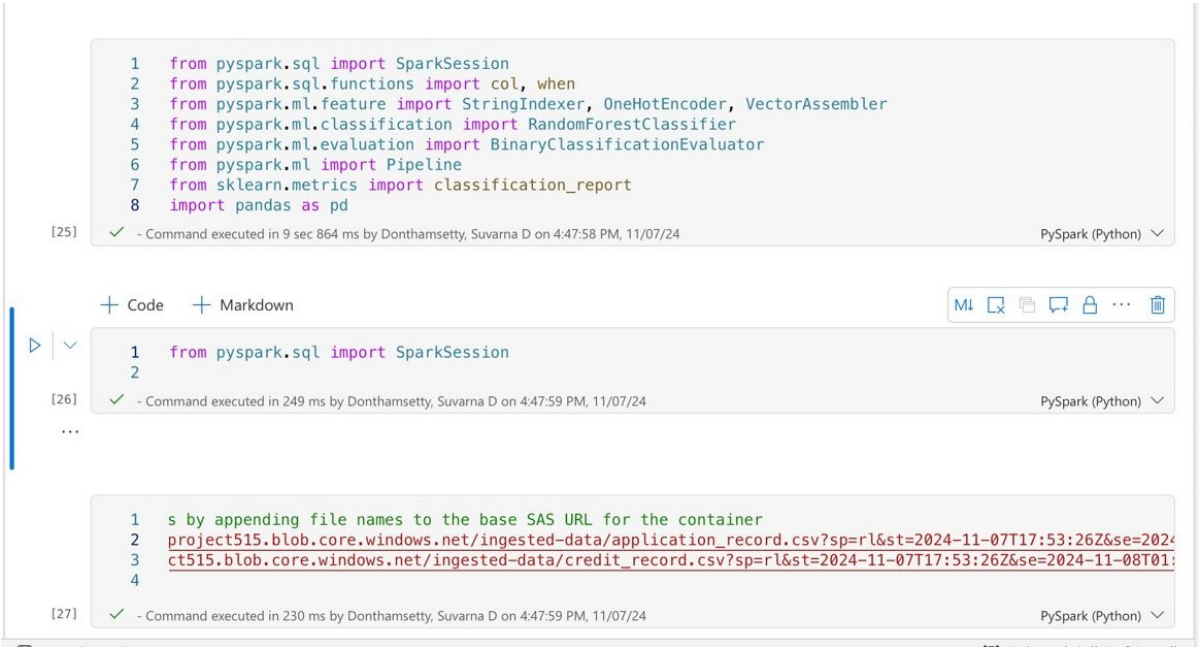

Figure. XVI.1

Figure. XVI.2

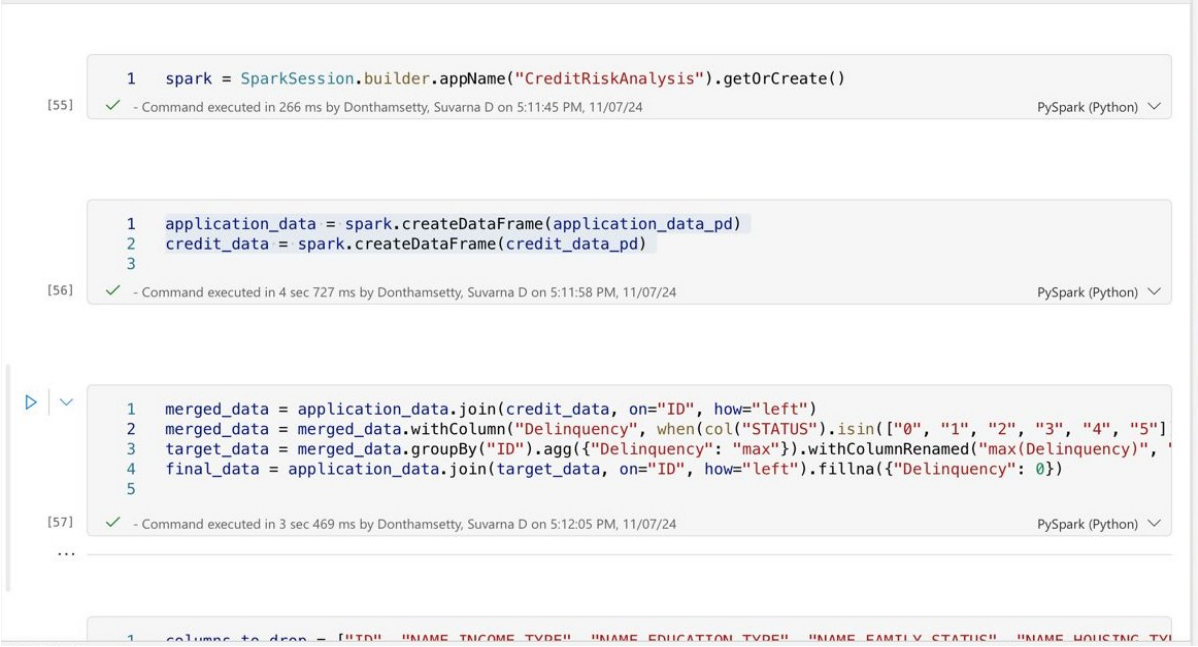

Figure. XVI.3

Figure. XVI.4

```
for col in categorical_cols:
    encoder = OneHotEncoder(inputCol=f"{col}_index", outputCol=f"{col}_ohe", handleInvalid="keep")
    final_data = encoder.fit(final_data).transform(final_data)

```

[61] ✓ - Command executed in 6 sec 342 ms by Donthamsetty, Suvarna D on 5:13:16 PM, 11/07/24 — PySpark (Python)

Run list | Run comparison | Customize columns

Run name	Start time	Duration	Status	Experiment na...
neat_drain_vkrzskgc	11/7/2024 5:13 PM	1s	Finished	Notebook-1
sharp_pig_6nxdhc9r	11/7/2024 5:13 PM	1s	Finished	Notebook-1
sweet_planet_7qzqnchd	11/7/2024 5:13 PM	1s	Finished	Notebook-1

Figure. XVI.5

```
train_data, test_data = final_data.randomSplit([0.7, 0.3], seed=42)
```

[63] ✓ - Command executed in 1 sec 510 ms by Donthamsetty, Suvarna D on 5:13:39 PM, 11/07/24 — PySpark (Python)

```
from pyspark.sql.functions import col
```

[67] ✓ - Command executed in 241 ms by Donthamsetty, Suvarna D on 5:15:04 PM, 11/07/24 — PySpark (Python)

```
majority_class = train_data.where(col("Delinquency") == 0)
minority_class = train_data.where(col("Delinquency") == 1)
ratio = majority_class.count() / minority_class.count()
oversampled_minority_class = minority_class.sample(withReplacement=True, fraction=ratio, seed=42)
balanced_train_data = majority_class.unionAll(oversampled_minority_class)

```

[69] ✓ - Command executed in 25 sec 272 ms by Donthamsetty, Suvarna D on 5:15:57 PM, 11/07/24 — PySpark (Python)

Figure. XVI.6

```
rf_model = RandomForestClassifier(featuresCol="features", labelCol="Delinquency", numTrees=100, maxDepth=1(
rf_model = rf_model.fit(balanced_train_data)
```

[70] ✓ - Command executed in 3 min 44 sec 340 ms by Donthamsetty, Suvarna D on 5:19:53 PM, 11/07/24 — PySpark (Python)

Output is hidden

```
predictions = rf_model.transform(test_data)
```

[71] ✓ - Command executed in 1 sec 500 ms by Donthamsetty, Suvarna D on 5:23:11 PM, 11/07/24 — PySpark (Python)

Figure. XVI.7

```
1  evaluator = BinaryClassificationEvaluator(labelCol="Delinquency", rawPredictionCol="rawPrediction", metricN
2  accuracy_rf = evaluator.evaluate(predictions)
3  print("Random Forest Model Accuracy (AUC):", accuracy_rf)
```

✓ - Command executed in 11 sec 994 ms by Donthamsetty, Suvarna D on 5:23:33 PM, 11/07/24 PySpark (Python)

```
Random Forest Model Accuracy (AUC): 0.8614004665215449
```

Figure. XVI.8

```
1  try:
2      y_test = test_data.select("Delinquency").toPandas()
3      y_pred_rf = predictions.select("prediction").toPandas()
4      print("Classification Report:\n", classification_report(y_test, y_pred_rf))
5  except Exception as e:
6      print("Error during conversion or classification report generation:", e)
```

[73] ✓ - Command executed in 19 sec 138 ms by Donthamsetty, Suvarna D on 5:24:34 PM, 11/07/24 PySpark (Python)

```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.89      0.86    121904
           1       0.88      0.83      0.86    122029

    accuracy                           0.86    243933
   macro avg       0.86      0.86      0.86    243933
weighted avg       0.86      0.86      0.86    243933
```

Figure. XVI.9

~End of Report~