**Adult Census Income Classification Project**

**1. Introduction**

The objective of this project is to predict whether an individual's annual income exceeds $50,000 based on various demographic and employment-related attributes. This binary classification task utilizes the Adult Census Income Dataset, also known as the "Census Income" dataset, sourced from the UCI Machine Learning Repository .

**2. Dataset Overview**

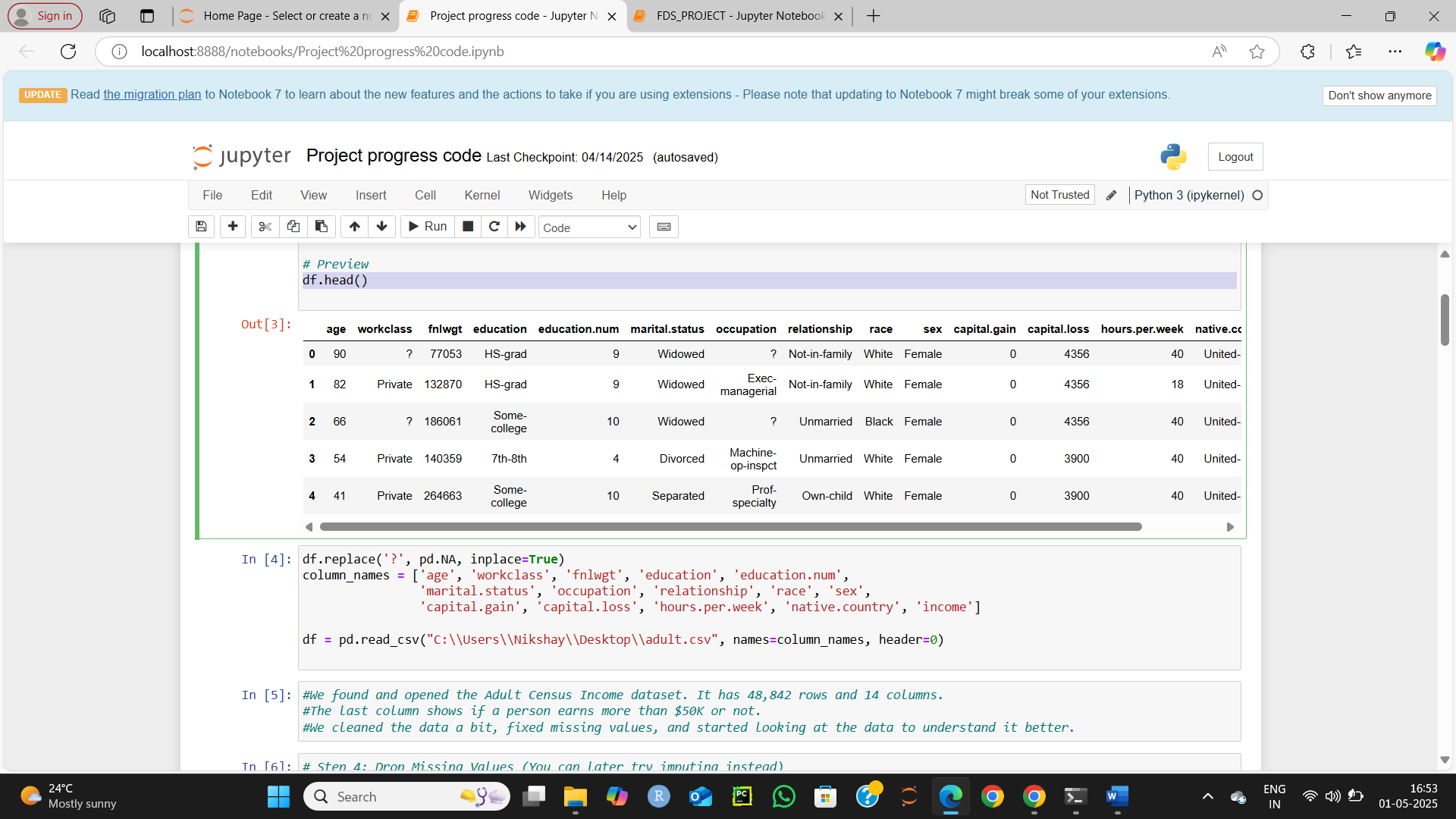
We used the Adult Census Income dataset from the UCI Machine Learning Repository, which contains 48,842 records with 14 features and a target variable called income, indicating whether a person earns more than $50K or not.

The features include:

Age, final weight (fnlwgt), capital gain, capital loss, and hours worked per week — all continuous.

Workclass, education, marital status, occupation, relationship, race, sex, and native country — all categorical.

Education-num is a numeric version of the education level.



**3. Data Preprocessing**

3.1. Handling Missing Values

Replaced all instances of '?' with NaN.

Dropped rows containing NaN values to ensure data integrity.

3.2. Encoding Categorical Variables

Utilized LabelEncoder from scikit-learn to convert categorical variables into numerical format.

3.3. Feature Scaling

Applied StandardScaler to standardize features by removing the mean and scaling to unit variance, essential for algorithms sensitive to feature scaling.

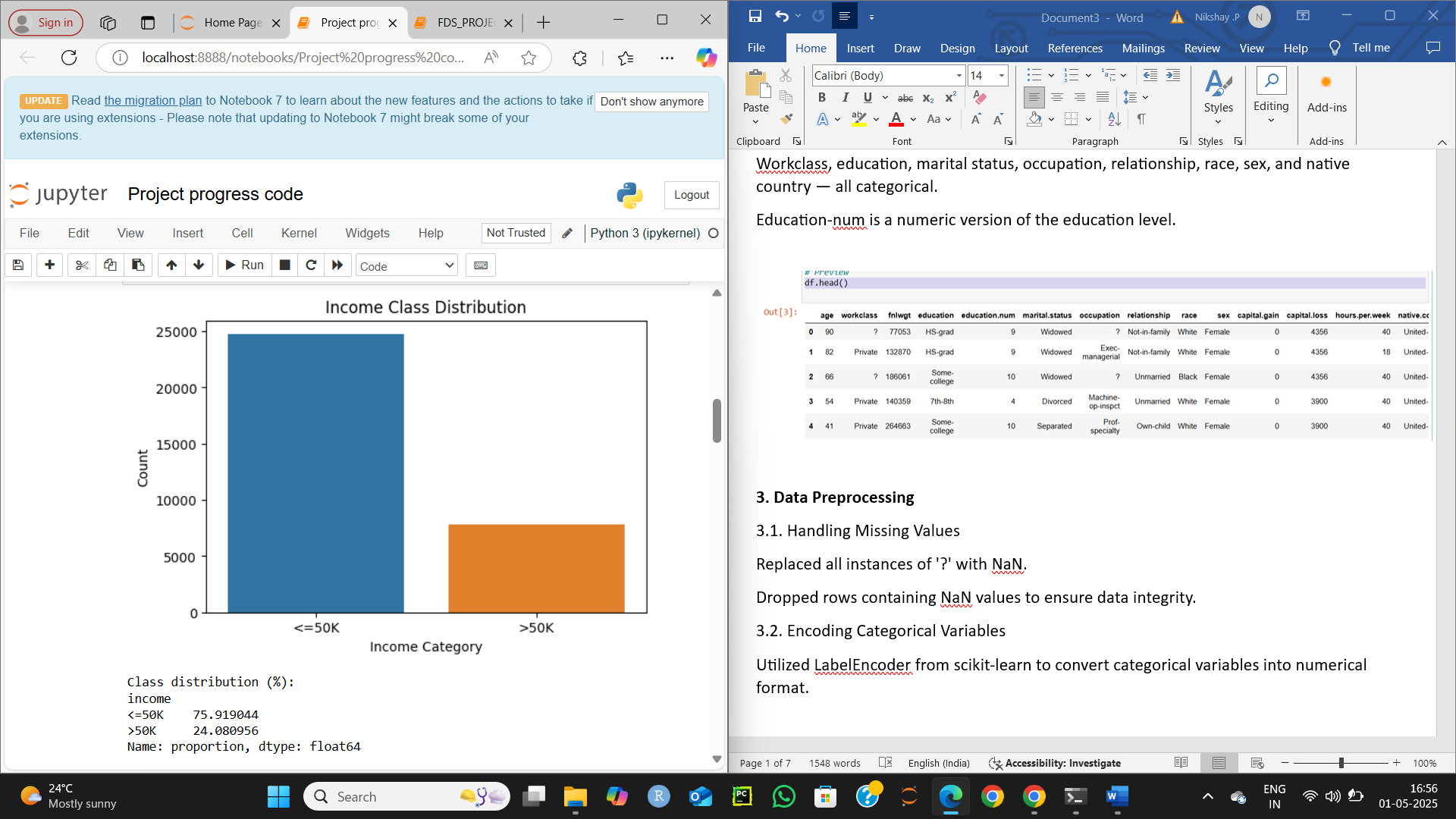
**4. Exploratory Data Analysis (EDA)**

4.1. Class Distribution

Income <=50K: ~75%

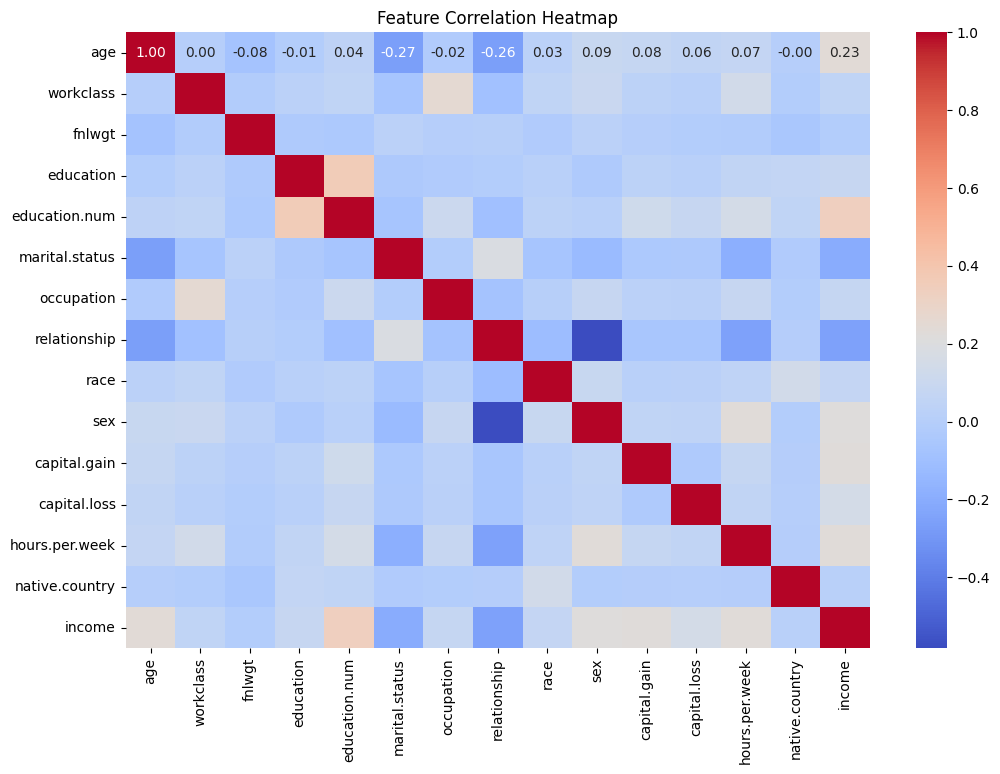
Income >50K: ~25%

This indicates a significant class imbalance, which can affect model performance.



4.2. Correlation Analysis

Generated a correlation heatmap to identify relationships between features.



Notable correlations:

education-num and income: Positive correlation

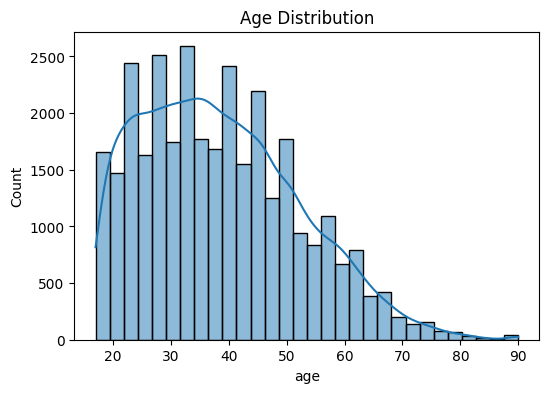
capital-gain and income: Positive correlation

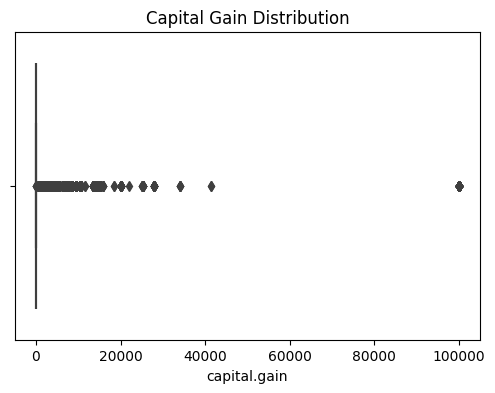
hours-per-week and income: Positive correlation

4.3. Feature Distributions

Age: Right-skewed distribution with a peak between 30-40 years.

Capital Gain: Highly skewed with many zeros and few high outliers.





**5. Dimensionality Reduction with PCA**

Applied Principal Component Analysis (PCA) to reduce dimensionality and visualize data:

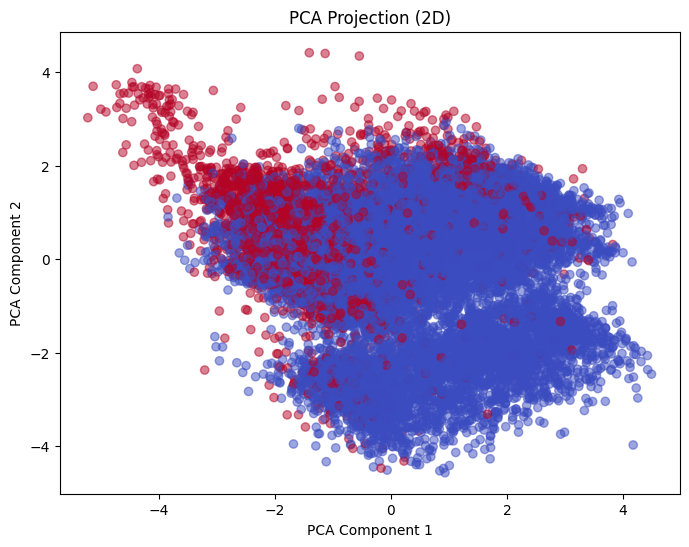
Explained Variance by Components:

Component 1: 14.9%

Component 2: 10.1%

Component 3: 8.2%

Total Variance Explained: ~33.2%



A 2D scatter plot using the first two principal components showed some separation between income classes, indicating that PCA can capture variance related to the target variable.

**6. Model Building and Evaluation**

6.1. Train-Test Split

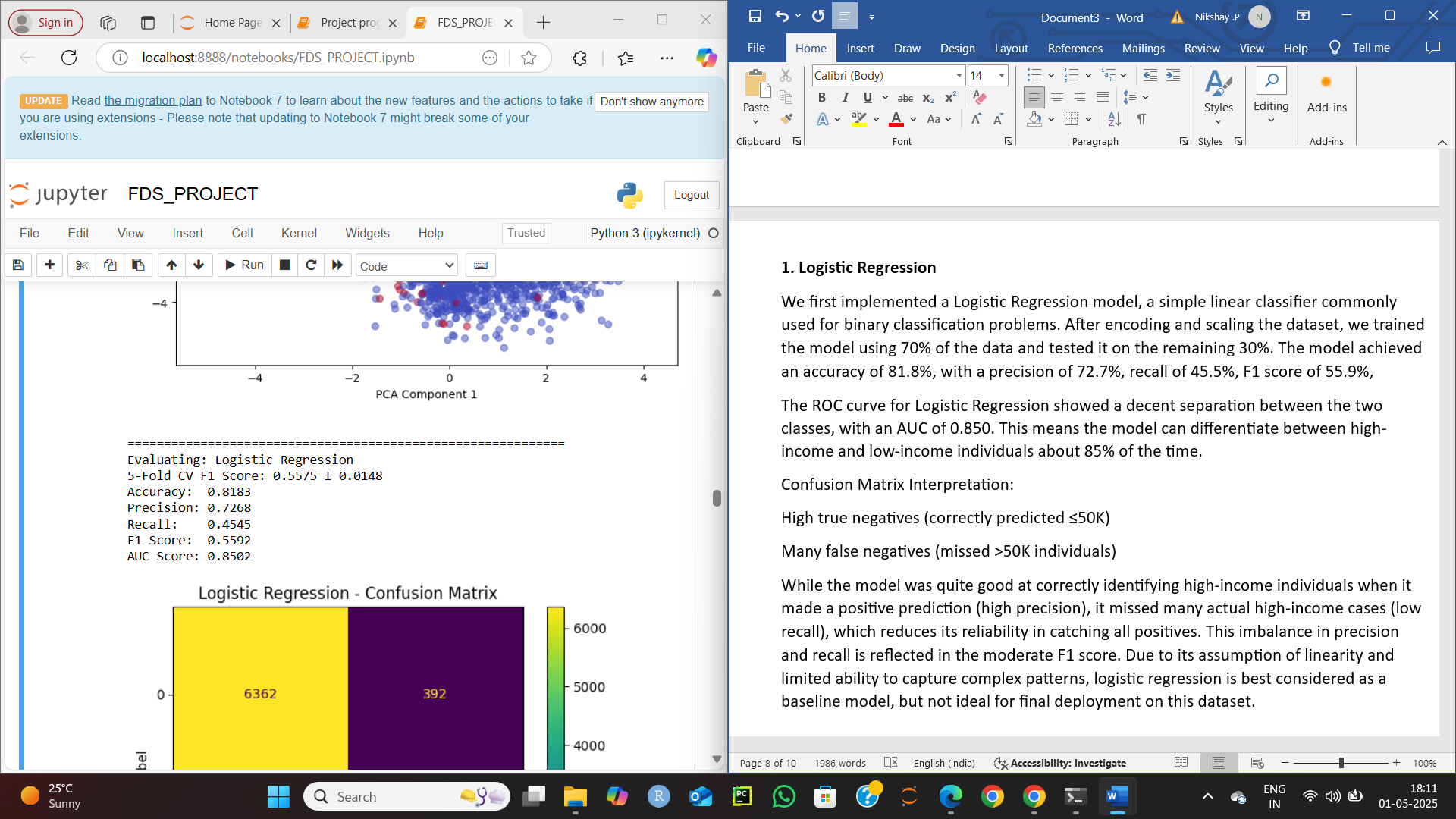
We divided the dataset into:

70% training data – used to train each model

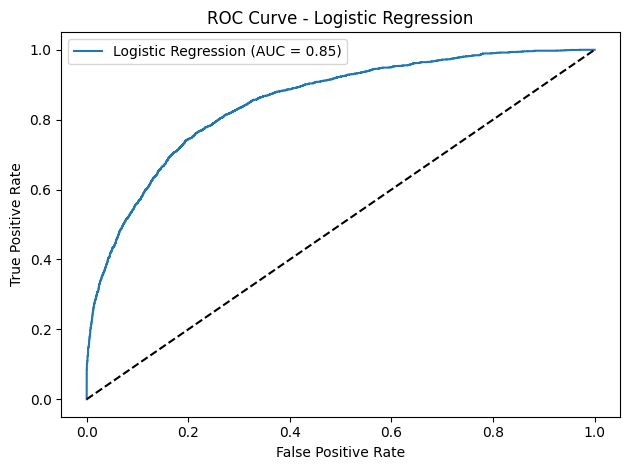
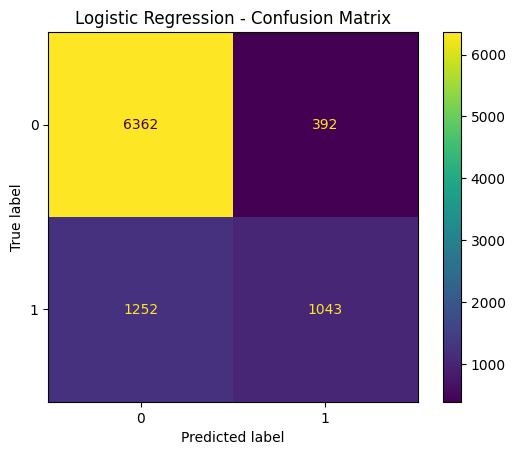
30% testing data – used to evaluate how well the trained models perform on unseen data

This helps us measure real-world performance and prevents overfitting.

**6.1. Logistic Regression**



We first implemented a Logistic Regression model, a simple linear classifier commonly used for binary classification problems. After encoding and scaling the dataset, we trained the model using 70% of the data and tested it on the remaining 30%. The model achieved an accuracy of 81.8%, with a precision of 72.7%, recall of 45.5%, F1 score of 55.9%,

The ROC curve for Logistic Regression showed a decent separation between the two classes, with an AUC of 0.850. This means the model can differentiate between high-income and low-income individuals about 85% of the time.

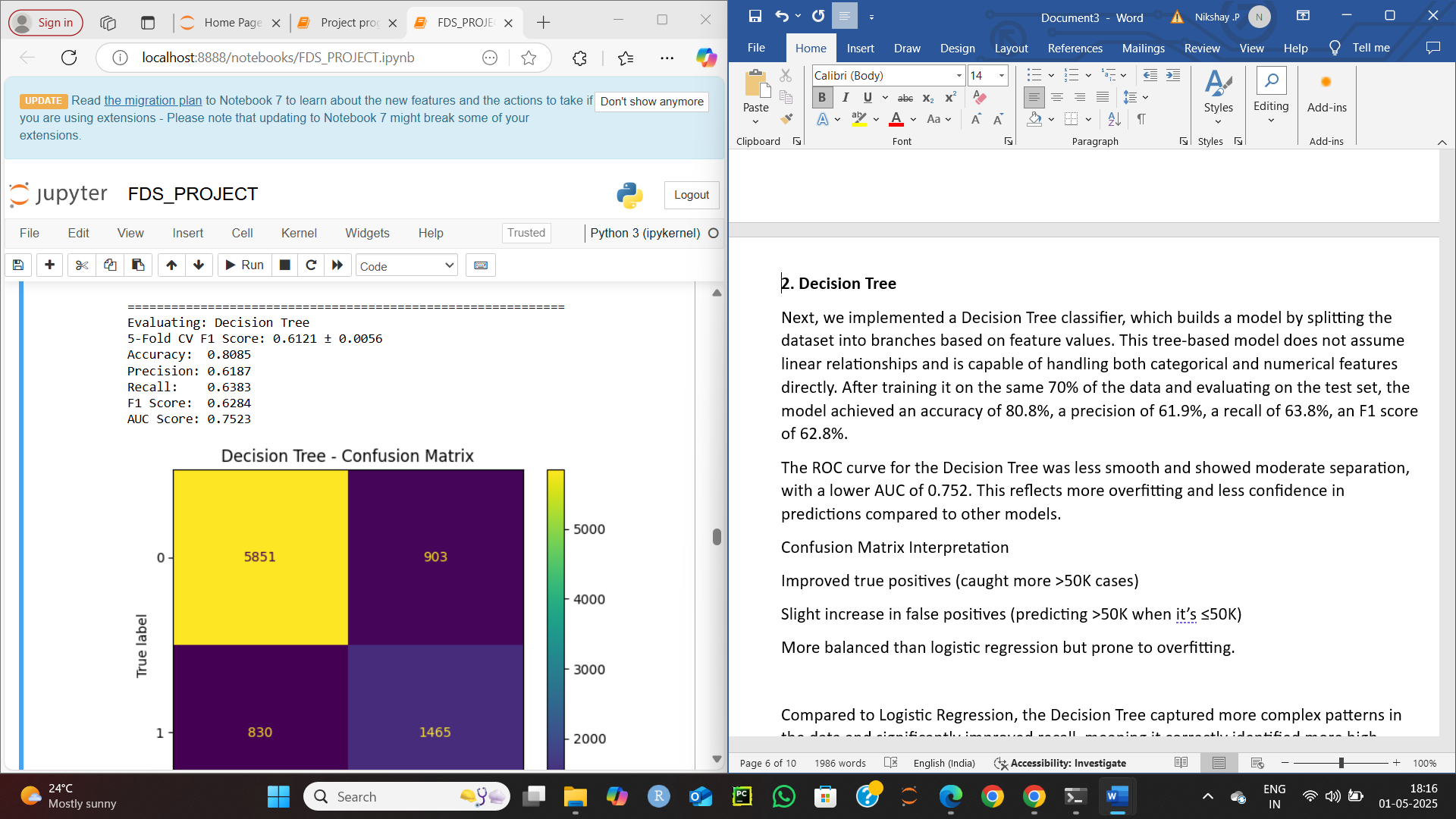
Confusion Matrix Interpretation:

High true negatives (correctly predicted ≤50K)

Many false negatives (missed >50K individuals)

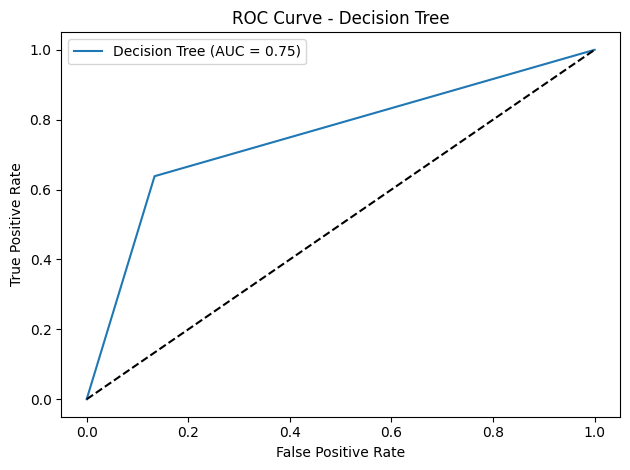
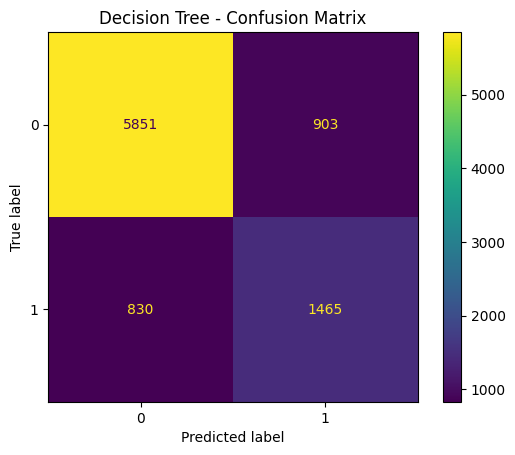
While the model was quite good at correctly identifying high-income individuals when it made a positive prediction (high precision), it missed many actual high-income cases (low recall), which reduces its reliability in catching all positives. This imbalance in precision and recall is reflected in the moderate F1 score. Due to its assumption of linearity and limited ability to capture complex patterns, logistic regression is best considered as a baseline model, but not ideal for final deployment on this dataset.

**6.2. Decision Tree**



Next, we implemented a Decision Tree classifier, which builds a model by splitting the dataset into branches based on feature values. This tree-based model does not assume linear relationships and is capable of handling both categorical and numerical features directly. After training it on the same 70% of the data and evaluating on the test set, the model achieved an accuracy of 80.8%, a precision of 61.9%, a recall of 63.8%, an F1 score of 62.8%.

The ROC curve for the Decision Tree was less smooth and showed moderate separation, with a lower AUC of 0.752. This reflects more overfitting and less confidence in predictions compared to other models.

Confusion Matrix Interpretation:

Improved true positives (caught more >50K cases)

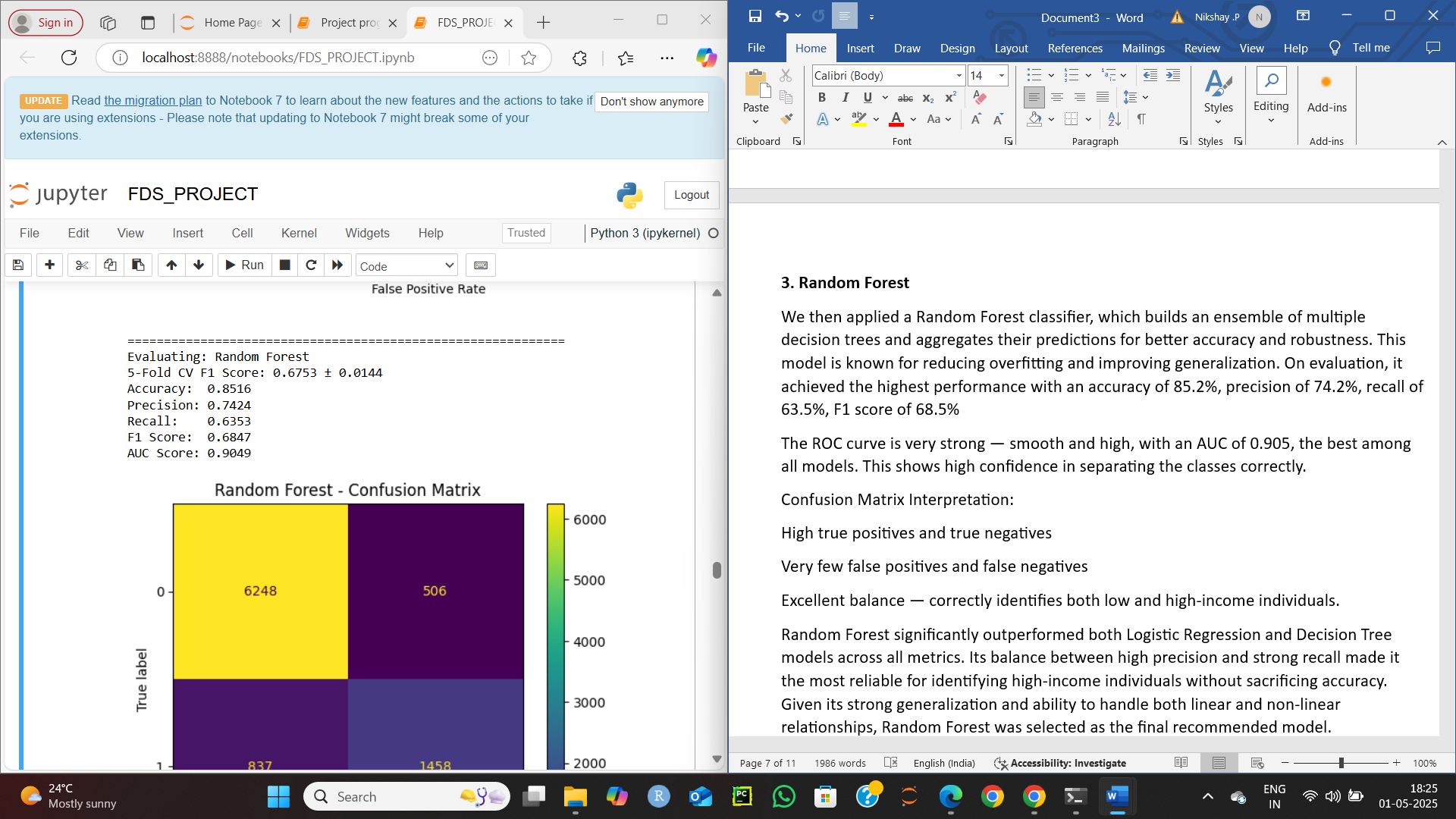
Slight increase in false positives (predicting >50K when it’s ≤50K)

More balanced than logistic regression but prone to overfitting.

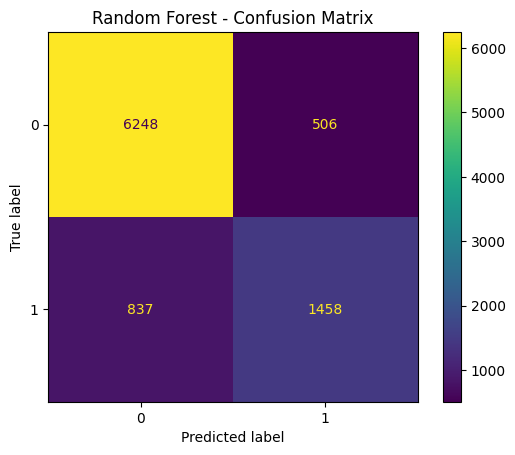
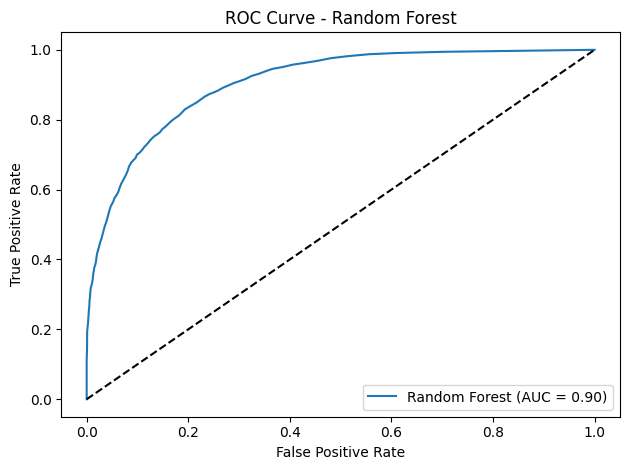
Compared to Logistic Regression, the Decision Tree captured more complex patterns in the data and significantly improved recall, meaning it correctly identified more high-income individuals. However, this came with a slight drop in overall precision and accuracy. Decision Trees are also known to overfit the training data if not carefully tuned, which can explain why its AUC score was lower than other models.

Overall, the Decision Tree provided a strong non-linear alternative to Logistic Regression, offering better recall and interpretability, but with limited generalization compared to more advanced ensemble methods.

**6.3. Random Forest**



We then applied a Random Forest classifier, which builds an ensemble of multiple decision trees and aggregates their predictions for better accuracy and robustness. This model is known for reducing overfitting and improving generalization. On evaluation, it achieved the highest performance with an accuracy of 85.2%, precision of 74.2%, recall of 63.5%, F1 score of 68.5%

The ROC curve is very strong — smooth and high, with an AUC of 0.905, the best among all models. This shows high confidence in separating the classes correctly.

Confusion Matrix Interpretation:

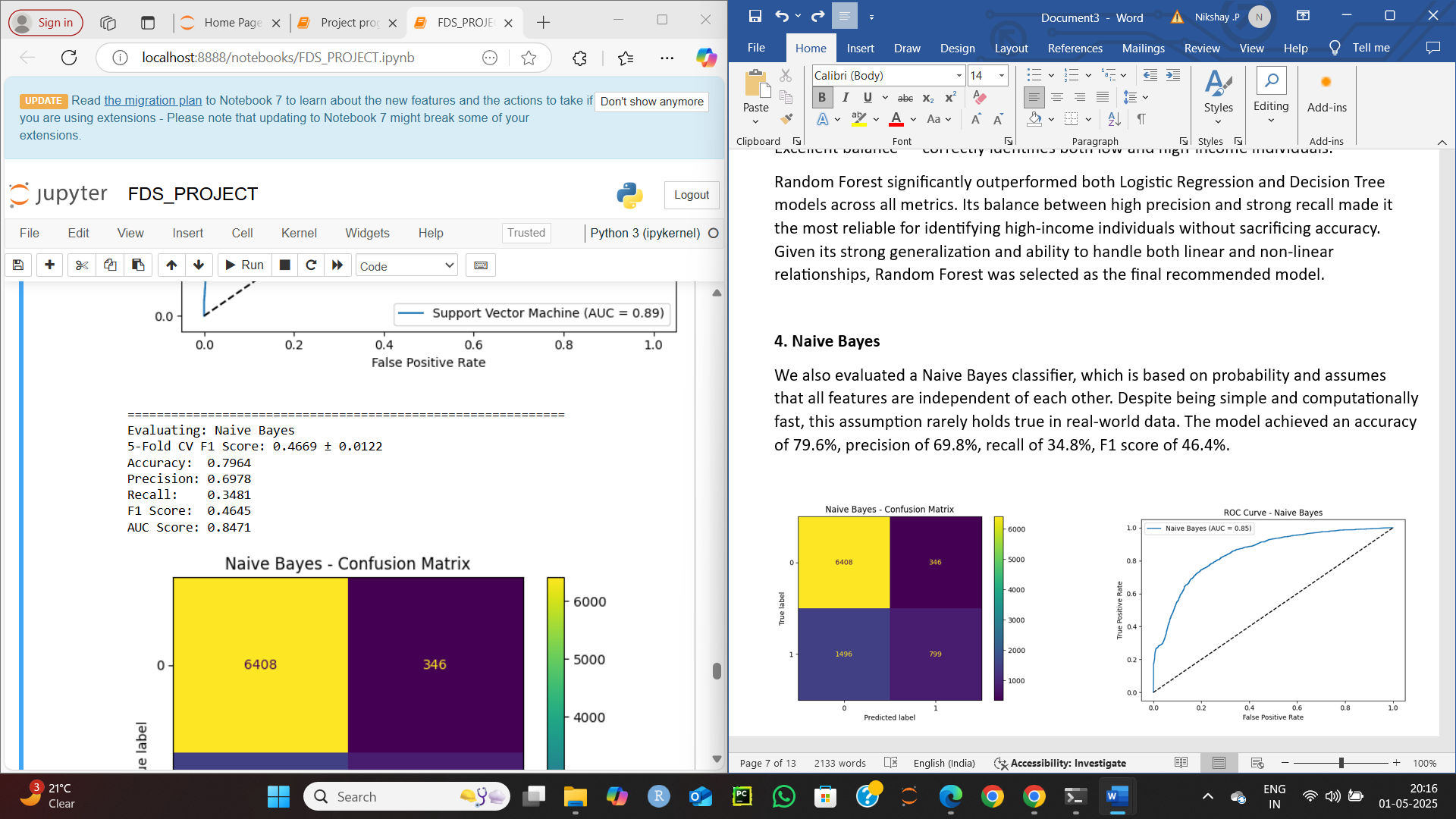
High true positives and true negatives

Very few false positives and false negatives

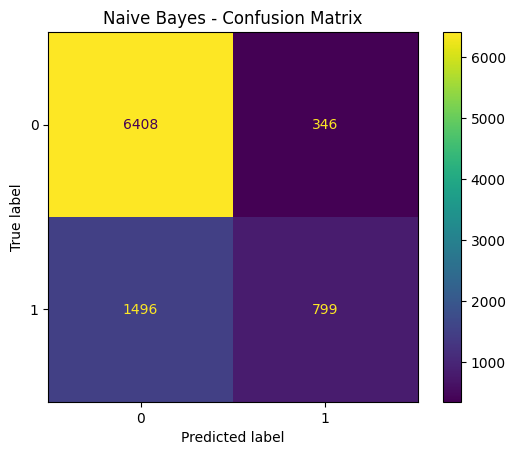
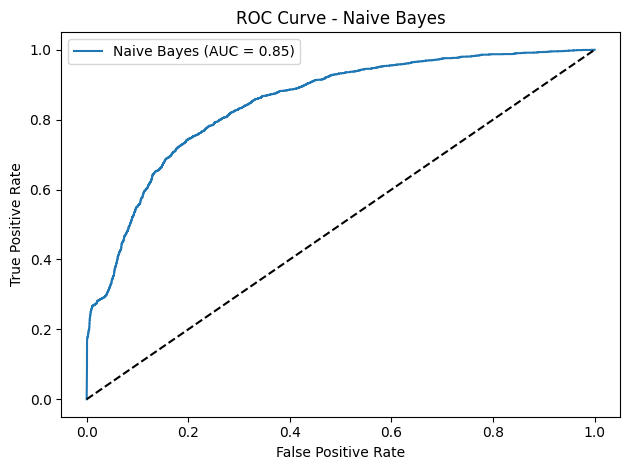
Excellent balance — correctly identifies both low and high-income individuals.

Random Forest significantly outperformed both Logistic Regression and Decision Tree models across all metrics. Its balance between high precision and strong recall made it the most reliable for identifying high-income individuals without sacrificing accuracy. Given its strong generalization and ability to handle both linear and non-linear relationships, Random Forest was selected as the final recommended model.

**6.4. Naive Bayes**



We also evaluated a Naive Bayes classifier, which is based on probability and assumes that all features are independent of each other. Despite being simple and computationally fast, this assumption rarely holds true in real-world data. The model achieved an accuracy of 79.6%, precision of 69.8%, recall of 34.8%, F1 score of 46.4%.

ROC Curve Interpretation:

AUC is decent (0.85), but the curve shows low recall zones — reflecting its inability to capture most >50K cases.

Confusion Matrix Interpretation:

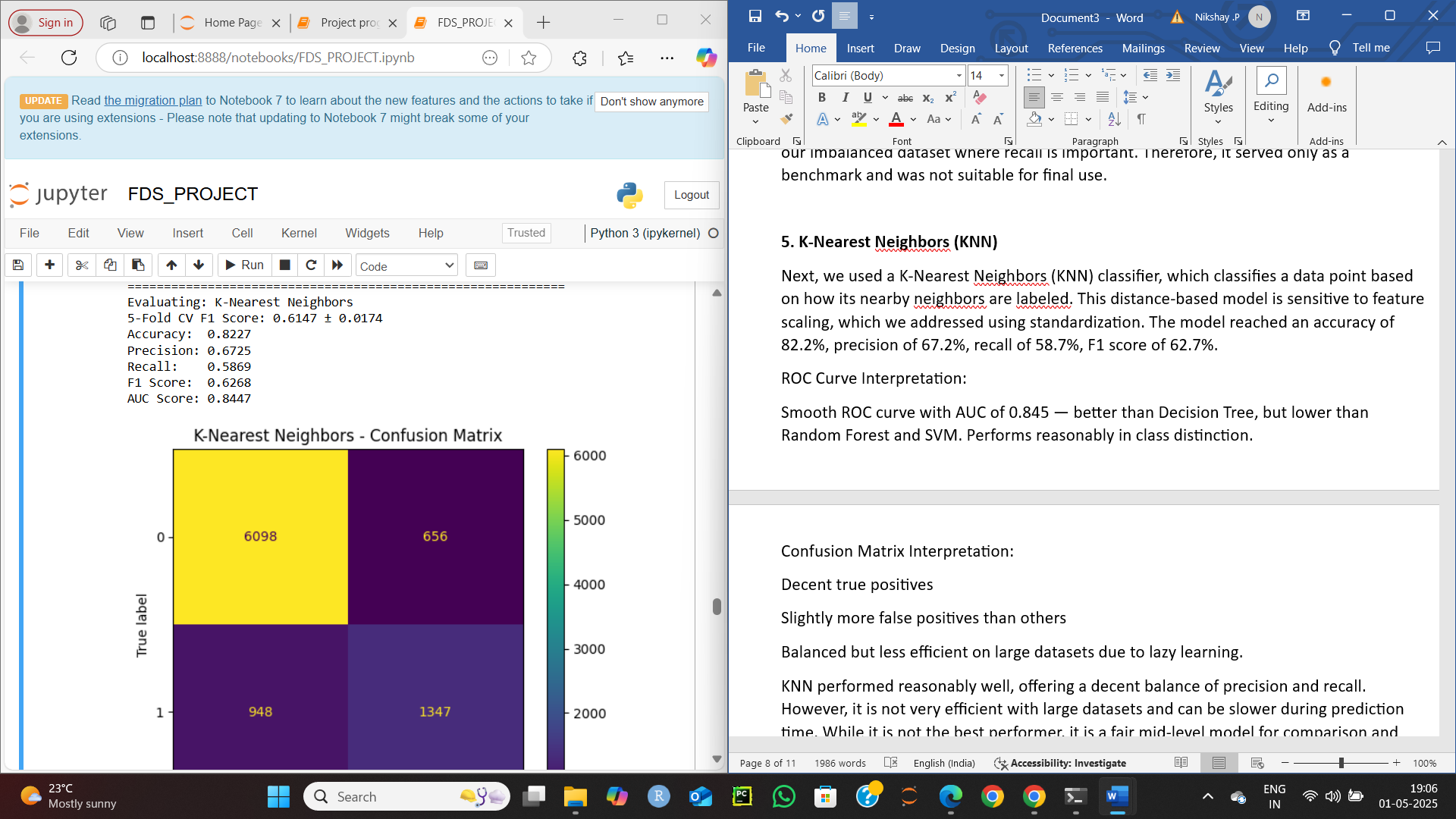
Many false negatives

Strong tendency to predict ≤50K, even when it’s >50K

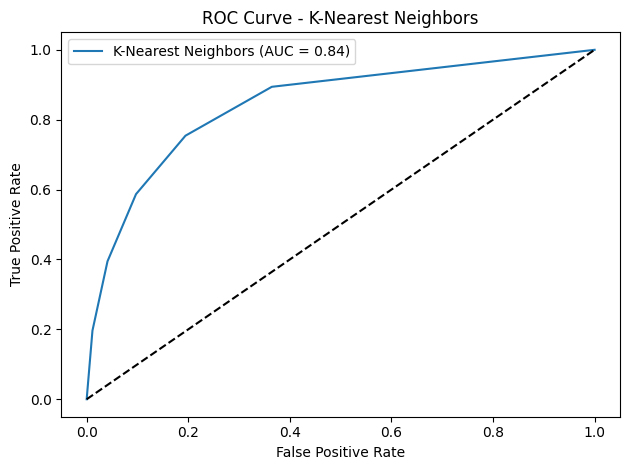
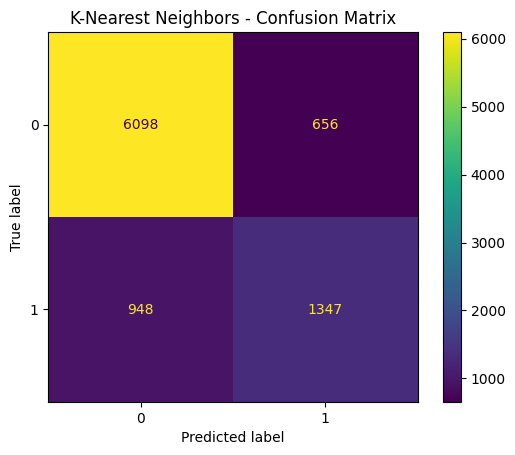
Very cautious model — underestimates high-income labels.

While precision was acceptable, the very low recall indicates that the model missed a large number of actual high-income individuals. This makes Naive Bayes a poor choice for our imbalanced dataset where recall is important. Therefore, it served only as a benchmark and was not suitable for final use.

**6.5. K-Nearest Neighbors (KNN)**



Next, we used a K-Nearest Neighbors (KNN) classifier, which classifies a data point based on how its nearby neighbors are labeled. This distance-based model is sensitive to feature scaling, which we addressed using standardization. The model reached an accuracy of 82.2%, precision of 67.2%, recall of 58.7%, F1 score of 62.7%.

ROC Curve Interpretation:

Smooth ROC curve with AUC of 0.845 — better than Decision Tree, but lower than Random Forest and SVM. Performs reasonably in class distinction.

Confusion Matrix Interpretation:

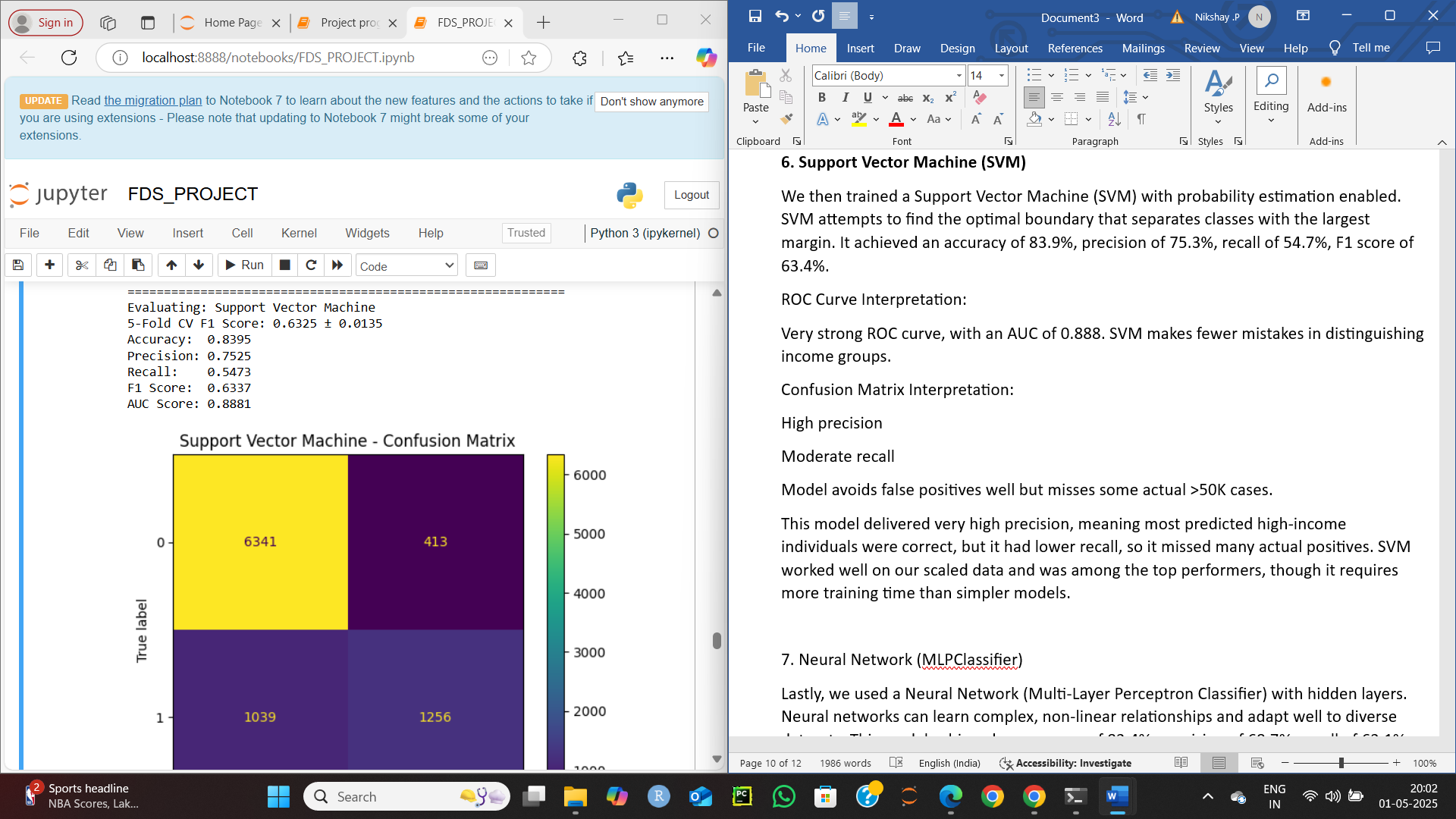
Decent true positives

Slightly more false positives than others

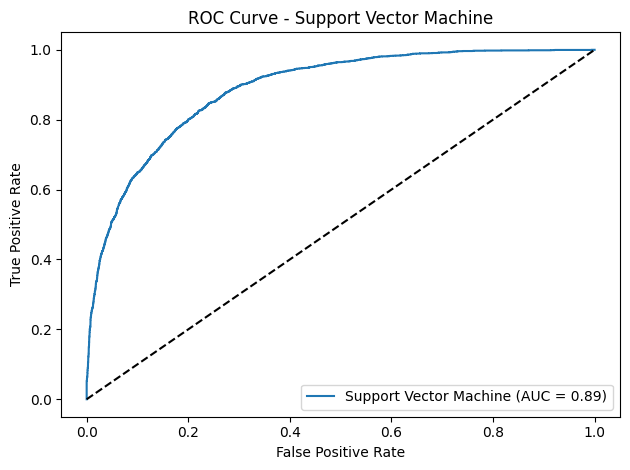
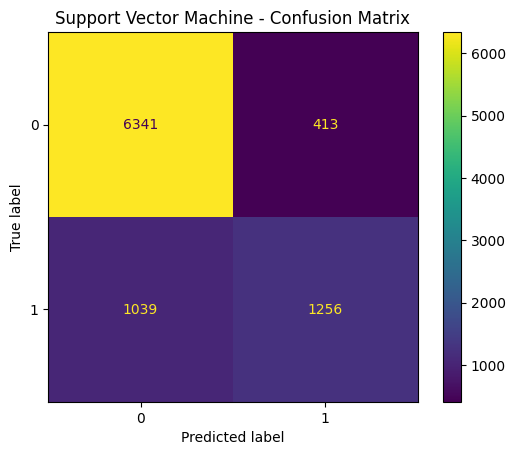
Balanced but less efficient on large datasets due to lazy learning.

KNN performed reasonably well, offering a decent balance of precision and recall. However, it is not very efficient with large datasets and can be slower during prediction time. While it is not the best performer, it is a fair mid-level model for comparison and showed good general behavior after scaling.

**6.6. Support Vector Machine (SVM)**



We then trained a Support Vector Machine (SVM) with probability estimation enabled. SVM attempts to find the optimal boundary that separates classes with the largest margin. It achieved an accuracy of 83.9%, precision of 75.3%, recall of 54.7%, F1 score of 63.4%.

ROC Curve Interpretation:

Very strong ROC curve, with an AUC of 0.888. SVM makes fewer mistakes in distinguishing income groups.

Confusion Matrix Interpretation:

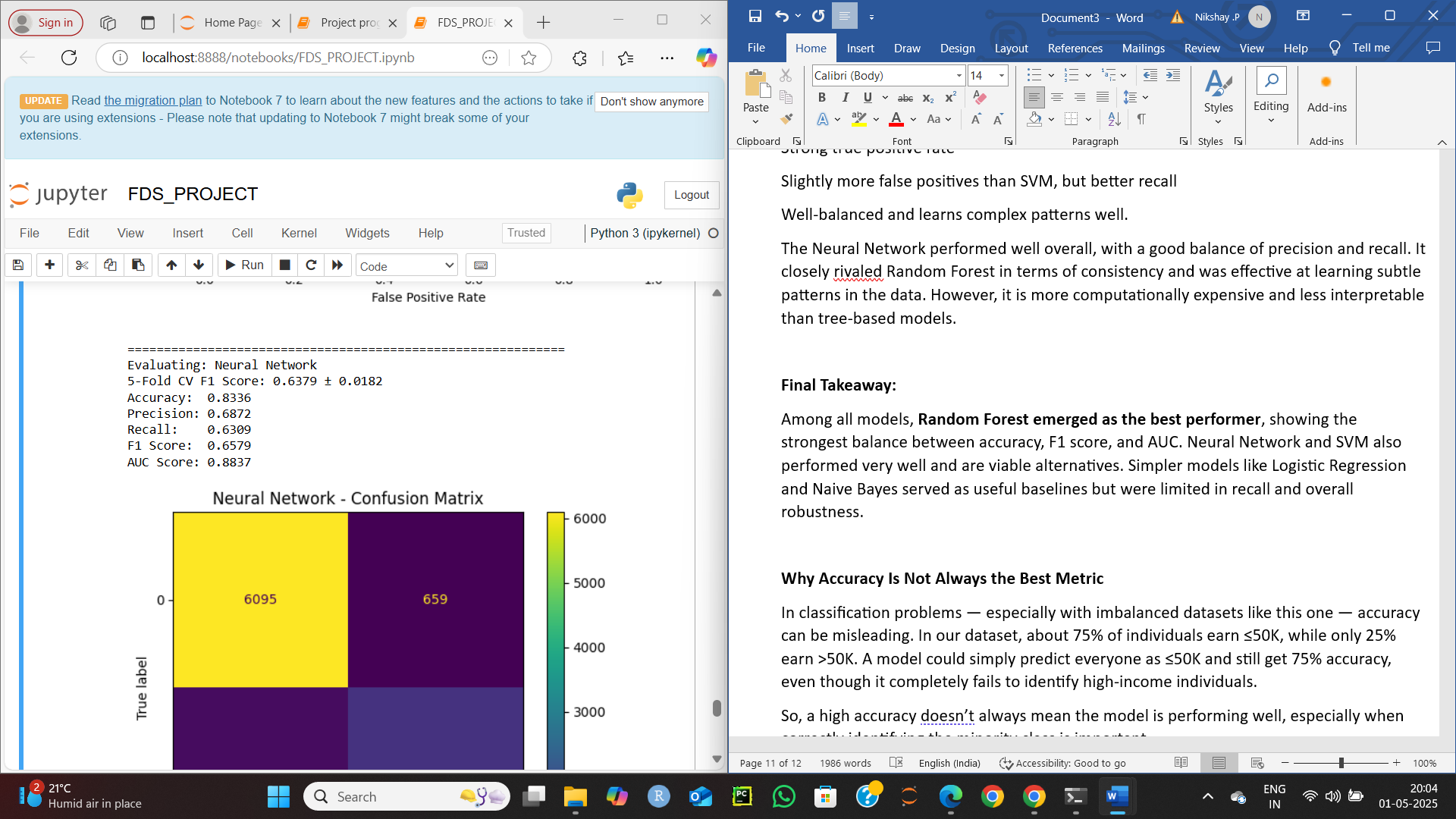
High precision

Moderate recall

Model avoids false positives well but misses some actual >50K cases.

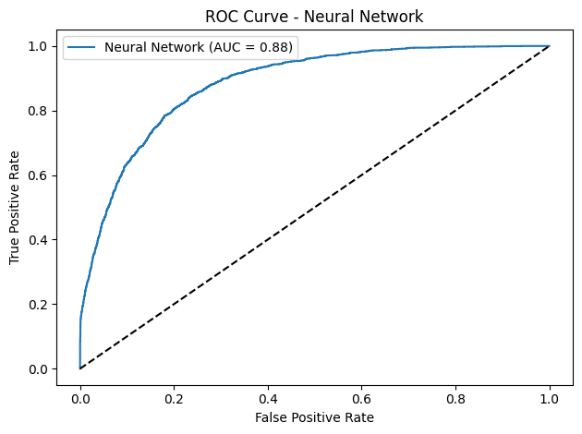
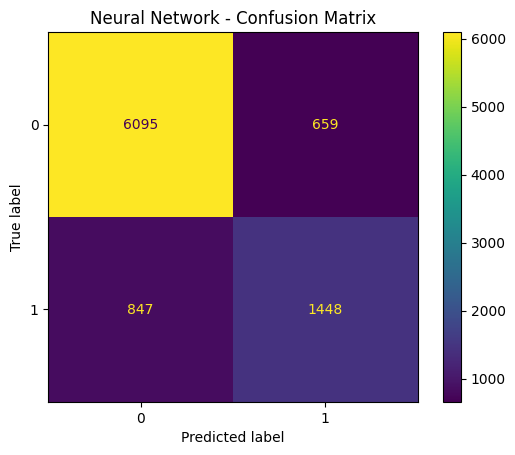
This model delivered very high precision, meaning most predicted high-income individuals were correct, but it had lower recall, so it missed many actual positives. SVM worked well on our scaled data and was among the top performers, though it requires more training time than simpler models.

**6.7. Neural Network (MLPClassifier)**



Lastly, we used a Neural Network (Multi-Layer Perceptron Classifier) with hidden layers. Neural networks can learn complex, non-linear relationships and adapt well to diverse datasets. This model achieved an accuracy of 83.4%, precision of 68.7%, recall of 63.1%, F1 score of 65.8%.

ROC Curve Interpretation:

AUC of 0.884 shows solid ability to distinguish classes. The ROC curve is high and consistent.

Confusion Matrix Interpretation:

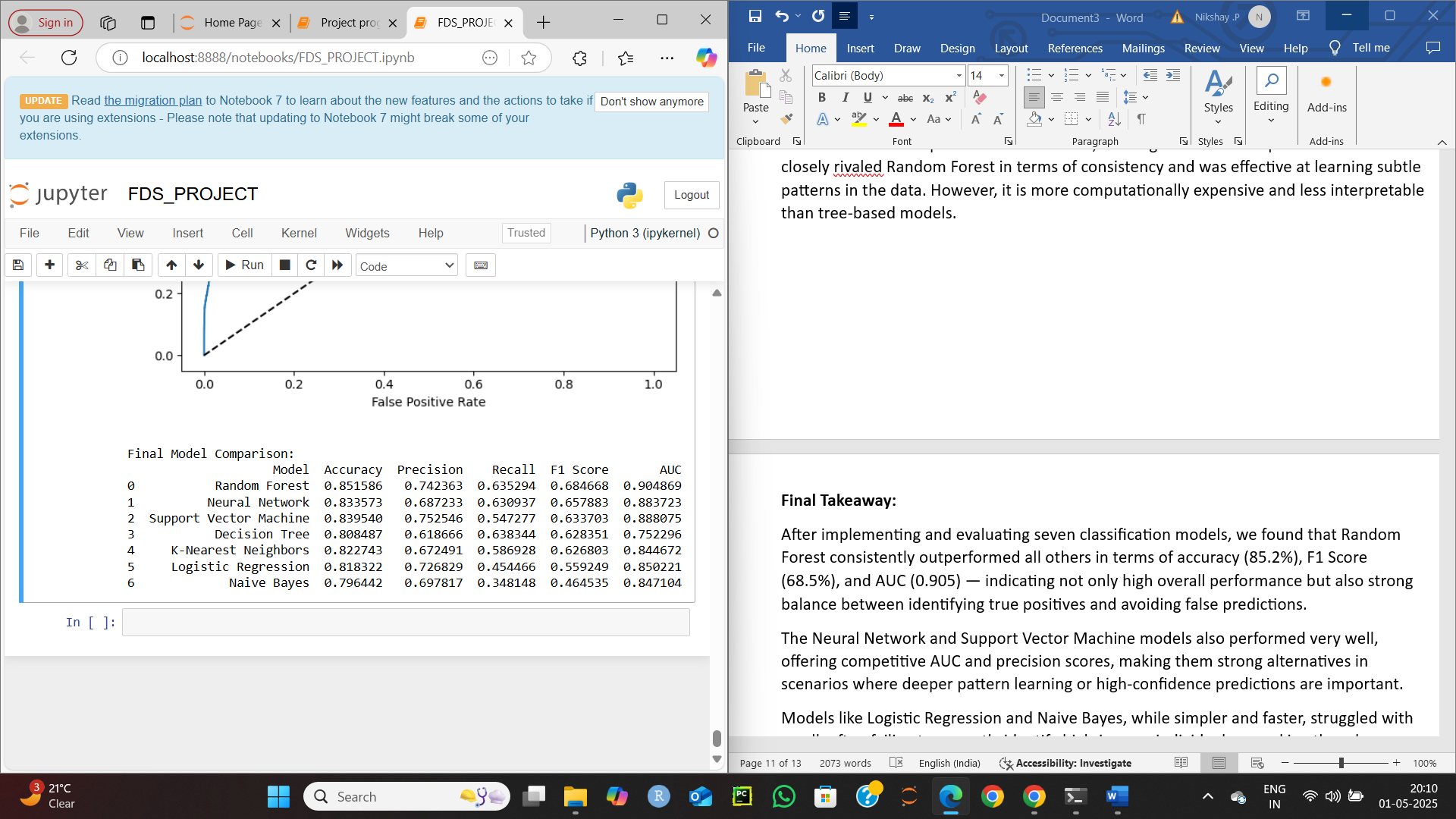
Strong true positive rate

Slightly more false positives than SVM, but better recall

Well-balanced and learns complex patterns well.

The Neural Network performed well overall, with a good balance of precision and recall. It closely rivaled Random Forest in terms of consistency and was effective at learning subtle patterns in the data. However, it is more computationally expensive and less interpretable than tree-based models.

**7.Final Takeaway:**



After implementing and evaluating seven classification models, we found that Random Forest consistently outperformed all others in terms of accuracy (85.2%), F1 Score (68.5%), and AUC (0.905) — indicating not only high overall performance but also strong balance between identifying true positives and avoiding false predictions.

The Neural Network and Support Vector Machine models also performed very well, offering competitive AUC and precision scores, making them strong alternatives in scenarios where deeper pattern learning or high-confidence predictions are important.

Models like Logistic Regression and Naive Bayes, while simpler and faster, struggled with recall, often failing to correctly identify high-income individuals — making them less suitable for this imbalanced classification task.

By focusing on F1 Score and AUC instead of just accuracy, we ensured fair and reliable model evaluation that reflects real-world needs where detecting the minority class (earning >50K) is critical.

**8.Why Accuracy Is Not Always the Best Metric**

In classification problems — especially with imbalanced datasets like this one — accuracy can be misleading. In our dataset, about 75% of individuals earn ≤50K, while only 25% earn >50K. A model could simply predict everyone as ≤50K and still get 75% accuracy, even though it completely fails to identify high-income individuals.

So, a high accuracy doesn’t always mean the model is performing well, especially when correctly identifying the minority class is important.

**9.Why We Used F1 Score**

To address this, we relied more on the F1 Score, which is the harmonic mean of precision and recall. This metric is especially useful when:

The data is imbalanced

We care about both false positives and false negatives

We want a single score that reflects the trade-off between precision and recall

**10.In our case:**

Precision tells us: When the model predicts a person earns >50K, how often is it correct?

Recall tells us: Of all people who actually earn >50K, how many did the model find?

The F1 Score combines both, helping us select models that are not just accurate, but also sensitive to both types of errors.

**11.How This Defines Our Project**

This project is focused on solving a real-world problem: predicting whether a person earns more than $50K per year using public census data. We approached it as a classification task and followed a full machine learning pipeline:

We started by analyzing the structure and imbalance of the data.

We performed data cleaning, encoding, and scaling to prepare for modeling.

We explored the data through EDA and PCA to better understand feature relationships.

We implemented and compared seven different machine learning models, from simple (Logistic Regression) to advanced (Neural Network), using F1 Score and AUC instead of just accuracy to guide our decisions.

By doing this, we not only built a predictive system, but we also showed the importance of choosing the right evaluation metrics when dealing with imbalanced classification problems — a mistake many models make in real life.

**12.Summary**

our project is a complete machine learning solution built on real census data. Its strength lies not only in model performance but in the thoughtful selection of evaluation metrics and techniques. By exploring various algorithms and emphasizing recall and F1-score over simple accuracy, we created a model that is both practical and insightful — showing how machine learning can be applied responsibly to real-world, imbalanced data.

**13.Future Importance**

To improve this project in the future, we plan to explore class balancing techniques like **SMOTE**, optimize model parameters using grid search, and experiment with more advanced ensemble methods such as **XGBoost.** Additionally, we aim to enhance interpretability through **SHAP**, deploy the best model as a web app, and apply richer feature engineering for better accuracy and fairness.