-REPORT FINAL PROJECT : Salary Prediction -

1. INTRODUCTION

This machine learning project centers around the classification of salaries, aiming to develop a model that categorizes individuals into two classes: those earning less than or equal to $50,000 and those earning more than $50,000 annually. The data set used for the development of the project comprises several features, such as demographic information, education, occupation, work hours among others to create an accurate salary classification model.

The project uses a supervised learning approach with classification algorithms as its core. To ensure an optimal result various models, such as logistic regression, decision trees and KNN will be explored and evaluated for their performance. A process of training and evaluation phases will be carried out to achieve a high degree of accuracy and reliability in predicting salary classes.

The project is expected to deliver a refined salary classification model that could be used in diverse real world fields such as financial institutions, human resources and loan approvals among others.

2. PROCESS AND METHODOLOGIES

● Selection and study of the database

We selected a salary dataset from Kaggle. The database was examined to understand its structure, features and variables. During the initial data exploration phase, we started by exploring the dataset's dimensions and handled the possible duplicated rows that could lead to biases in the model. We then observed that the dataset contained missing values (?) in certain instances. We replaced the "?" string with NaN and we finally dropped all of them → We had the privilege to deal with the NaNs by just droppin them as the size of our dataset was big enough. (32561 rows).

Once we dropped all the missing values we computed several graphics to visualize the relationships and characteristics of our data. We plotted a workclass count by salary range which clearly showed how the private sector dominates in the <= 50K salary as well as the Bachelors and Some-college degrees in the Education Level Count by Gender graphic.

● Modification of categorical values

After completing the first analysis we saw that our dataset was composed of several categorical values which had to be transformed to numerical values in order to work with them.

At first, we decided it would be suitable to handle these categorical variables by just assigning a numerical value to all of them without treating them as binary situations and labeling from 0 to the number of existing variables. We realized this approach was not optimal as the fact of assigning random values and not treating them as binary (0 or 1) affected the correlation between them.

Even though it is a quite laborious method we decided to solutionate this inconvenience by treating all the possible values in every variable as a binary situation. For example, regarding the Country feature, Canada could get a 0 value (not from Canada) or 1 value (from Canada).

- ## Outliers

In this part of the process it was crucial to choose an accurate threshold value in order to determine the number of rows that were going to be considered as outliers.

- A smaller threshold would be less likely to flag data points as outliers, but it would be more likely to miss some true anomalies.
- On the other hand, a larger threshold would be more sensitive to outliers but may also flag more false positives.

We visually inspected how our data changes with different values of thresholds to see the ranges of each feature and see if they make sense inside the desired values. We also had to ensure that the outliers had been removed successfully.

Finally, we concluded that 5 was a better threshold than 1.5 because the latter removed too many rows. Even though 5 is still removing a lot of them, usually the outliers are not that much. We then decided to evaluate similar higher values.

- ## Data normalization (0,1)

In order to have all the data in the same scale and avoid numerical dominance or instabilities we decided to normalize our data with a Z-score where the features would have a mean of 0 and a standard deviation of 1.

Once we had all the normalized data we computed a correlation matrix to know the relationships between the features. This information would help us to later select the most important ones.

- ## Feature Selection

To select the important features we used the PCA technique. By applying this method we reduced the dimensionality of the data set transforming the original features into this smaller set of variables of principal components.

We computed the variance ratio in order to understand how much information each of the selected principal components carried and choose the ones which explained a higher percentage of the total variance while reducing the dataset.

- Split of data: training, validation and test set / Training process

Data splitting is a crucial step in the preparation for the training process of a classification model. The objective was to create distinct sets of data that have its purposes, such as training the model, validating its performance and testing how it generalizes. We used different sets to see the differences between them:

Unbalanced Original:

- *X_train:* Features for training the model on the unbalanced original dataset.
- *y_train:* Corresponding labels for the training set.
- *X_val:* Features for validating the model on the unbalanced original dataset.
- *y_val:* Corresponding labels for the validation set.

Balanced Original:

- *X_train_balanced:* Features for training the model on a balanced version of the original dataset. This balancing could involve techniques such as oversampling minority classes or undersampling majority classes.
- *y_train_balanced:* Corresponding labels for the balanced training set.
- *X_val:* Features for validating the model on the balanced original dataset.
- *y_val:* Corresponding labels for the validation set.

Unbalanced PCA:

- *X_train_pca:* Features for training the model on the unbalanced dataset after applying Principal Component Analysis (PCA). PCA is a dimensionality reduction technique.
- *y_train:* Corresponding labels for the training set.
- *X_val_pca:* Features for validating the model on the unbalanced dataset after PCA.
- *y_val:* Corresponding labels for the validation set.

Balanced PCA:

- *X_train_pca_balanced*: Features for training the model on a balanced version of the dataset after applying PCA.
- *y_train_balanced:* Corresponding labels for the balanced training set.
- *X_val_pca*: Features for validating the model on the balanced dataset after PCA.
- *y_val:* Corresponding labels for the validation set.

When writing the code we realized it was more optimal and efficient to do a modular and organized approach using functions for each model and apply it to the four different sets.

- Models

We decided to test 6 different models: Random forest, decision tree, XGB, Gradient boost, KNN and Logistic regression.

One problem we encountered during the process was the use of identical names for models across different datasets. This introduced a complication as, upon execution, the data from the last dataset was overwriting the previous ones. Consequently, only the results from the final dataset were saved, rendering the accumulated results meaningless.

Once we changed the names and solved the problem, in order to visually compare the different models we computed a precision-recall curve or ROC curve depending if the data set was balanced or not.

→ We chose the Precision-Recall curve when the class distribution was unbalanced and we were concerned about the false positives.

→ We used the ROC curve to illustrate the performance of the models in the balanced sets. The area under the ROC curve (AUC) is the parameter used to quantify the overall performance of the model.

Also we did a post-training process where we created a dataframe displaying the results of accuracy for each class model comparing the “normal” accuracy and the balanced accuracy. We implemented a weighting mechanism to ensure a more equitable assessment of the model's performance across different classes.

Based on the presented results for both non-PCA and PCA scenarios we have extracted these conclusions:

Random Forest:

- In the non-PCA scenario, Random Forest exhibits relatively consistent performance across balanced and unbalanced datasets.
- In the PCA scenario, Random Forest maintains a similar trend, with slightly reduced accuracy in the balanced dataset.

Random Forest:

- In the non-PCA scenario, Random Forest exhibits relatively consistent performance across balanced and unbalanced datasets.

- In the PCA scenario, Random Forest maintains a similar trend, with slightly reduced accuracy in the balanced dataset.

Decision Tree:

- Decision Tree shows competitive performance in the non-PCA balanced dataset.
- However, in other scenarios, especially in the PCA scenarios, its performance diminishes.

XGBoost (XGB):

- XGBoost consistently performs well across all scenarios, demonstrating its robustness to different dataset configurations.

Gradient Boosting:

- Gradient Boosting tends to perform better in the non-PCA scenarios, with a notable accuracy increase in the balanced dataset.

K-Nearest Neighbors (KNNC):

- KNNC shows mixed results, with varying performance across different scenarios. Notably, it exhibits competitive accuracy in the non-PCA unbalanced dataset.

Logistic Regression:

- Logistic Regression demonstrates stable performance across different scenarios, with slightly higher accuracy in the PCA scenarios.

Considering the overall performance, **XGBoost stands out** as a robust performer across balanced and unbalanced datasets, in both PCA and non-PCA scenarios.

- Voting

In order to improve the overall predictive performance and robustness of the model we decided to implement a voting system:

→ Hard voting: each classifier in the ensemble makes a prediction, and the final prediction is determined by a majority vote. The class that receives the majority of votes is chosen as the final prediction.

→ Soft voting: each classifier provides a probability estimate for each class, and the final prediction is based on the average of these probabilities. We calculated the normalized weights depending on the AUC: as bigger the area as bigger the weight.

3. CONCLUSIONS

In conclusion, we have developed a classification model by testing in different sets (balanced, unbalanced, PCA and not PCA) the 6 selected models (Random Forest, Decision Tree, XGBoost, Gradient Boosting, KNNC and Logistic Regression).

In the data preprocessing phase, we analyzed the data set including addressing missing values, managing outliers and handling the categorical values. Then, we decided to apply the principal component analysis (PCA) to reduce the dimensionality and focus on essential features while the Z-score normalization ensured a consistent data scale.

Once we completed all these steps we decided to test in 6 different models across various scenarios - non PCA, PCA, balanced and unbalanced data sets. Precision-recall curves and

ROC curves were used for model comparison as well as the voting system where we could see that the overall best option was XGB.

With this project we have developed a pretty accurate salary classification model which predicts if an individual has a lower or higher than 50K salary. We realized the importance of addressing data challenges and testing the model in different scenarios in order to improve and compare the results and get the best option to work with.