COMPARATIVE ANALYSIS OF SVM, DECISION TREE, AND RANDOM FOREST ALGORITHMS ON UCI CENSUS INCOME DATASET

Student ID:	**21062644**
Github Project Link:	https://github.com/faadeola/uci-census-income-classification

INTRODUCTION

This report compares the performance of three classification models, namely Support Vector Machine (SVM), Decision Tree Classifiers, and Random Forest Classifier. The goal is to assess the effectiveness of these models in classifying outcomes when applied to a real-world dataset like the UCI census income dataset. The UCI dataset, consisting of approximately 32,000 records and 14 attributes, was selected for its multivariate data characteristics, such as the combination of categorical (e.g., occupation, education) and numerical (e.g., age, hours-per-week) features. The classification task is to determine if an individual's annual income is greater than $50,000 or less than or equal to $ 50,000.

METHODOLOGY AND DATA PREPROCESSING

To prepare the dataset for model training and the classification task, I have taken the following preprocessing steps:

1. Removed rows with missing values using the df.dropna() and outliers from the dataset.
2. Stripped unnecessary whitespace in the categorical columns using the str.strip() from pandas (*Pandas Developers, n.d.*).
3. Replaced the target feature (Income) to have values of 0 for income <=50K and 1 for income>50K using the np.where function (*Numpy developers, n.d.*).
4. Removed duplicate values in the data using the df.drop_duplicates() pandas function.
5. Categorical features were transformed using OneHotEncoder(), while the numerical features were scaled using the StandardScaler().
6. Due to the class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was introduced to generate new synthetic sample data from the minority class, thereby addressing class imbalance (*Dodwell, n.d.*).
7. Separated the dataset into 14,756 rows (80%) for training and 3,689 (20%) for testing.
8. Streamlined preprocessing and model selection steps using the imblearn pipeline module for consistency.

MODEL COMPARISON AND METRICS

The models were trained using their default parameters from sklearn, except for the SVM. For the SVM, the Radial Basis Function (RBF) kernel was used, which helps transform non-linear data, as shown in the pairplot diagram, into a higher-dimensional decision space for better separation. Additionally, the regularization parameter (C) was set to 5 to control the trade-off between minimizing training error and testing error. After fitting the models on the training data and making predictions on the test split, the following results summary was obtained:

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC	Accurate pred
SVM	80.5%	85%	81%	82%	80%	2,971
Decision Tree	78.6%	80%	79%	79%	70%	2,900
Random Forest	82.8%	83%	83%	83%	75%	3,056

Figure 1: Summary table for model performance metrics

In the table above, SVM has the highest ROC-AUC score (80%), indicating better model performance in distinguishing between classes. Random Forest achieves the highest accuracy score (~83%) and the highest number of accurate predictions (3,056 out of 3,689), out-performing both SVM and Decision Tree (Worst-performer).

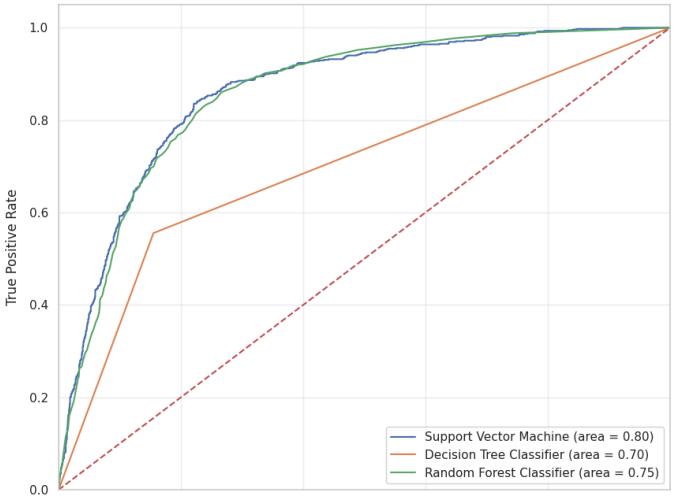

Figure 2: ROC Curve visualization for the three models

CONCLUSION

Based on the models' evaluation metrics, SVM shows its ability to differentiate between two classes. However, the Random Forest Classifier appears to be the better choice in this case due to its higher accuracy and consistent performance across all metrics. Further improvements could be made by fine-tuning the models' hyper-parameters using techniques like GridSearchCV, Cross Validation, etc. or experimenting with different feature engineering methods (such as PCA or RFE) could enhance the model's performance beyond the default settings.

REFERENCES

1. Dodwell, T. (2023, April 28). Rebalancing your data with the Synthetic Minority Oversampling Technique (aka SMOTE). digiLab. https://www.digilab.co.uk/posts/synthetic-minority-oversampling-technique
2. Google Developers. (n.d.). ROC and AUC. Google. Retrieved March 26, 2025, from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc
3. Kohavi, R. (1996). Census Income [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5GP7S.
4. NumPy developers. (n.d.). *numpy.where*. NumPy. Retrieved March 22, 2025, from https://numpy.org/doc/2.2/reference/generated/numpy.where.html
5. Pandas Developers. (n.d.). *pandas.Series.str.strip*. Pandas Documentation. Retrieved March 26, 2025, from https://pandas.pydata.org/docs/reference/api/pandas.Series.str.strip.html
6. Plagata, T. (2020, November 17). How scikit-learn pipelines make your life so much easier. Medium. https://tiaplagata.medium.com/how-scikit-learn-pipelines-make-your-life-so-much-easier-3cfbfa1d9da6
7. Scikit-learn developers. (n.d.). sklearn.compose.ColumnTransformer. Scikit-learn. Retrieved March 23, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html