Salary Classification Model

Franko Ndou

Business Problem

Microsoft is tasking me with demonstrating my data science expertise by assisting their HR department in developing a classification model. This model will predict whether job applicants are likely to earn more or less than $50,000 annually, helping Microsoft optimize its hiring process for a specific $50,000 annual salary position. By successfully developing this data-driven solution, I will not only contribute to improving their HR practices but also showcase my capabilities as a data scientist. Microsoft's commitment to innovation and efficiency in talent acquisition aligns with this project's goal.

Microsoft

Dataset

- I got my dataset from UCI Machine Learning Repository. It is salary census data from 1994.
- The dataset was imbalanced so I had to SMOTE it to be able to properly create my classification Models
- The target variable within the dataset is "income" which tells us if an individual makes Above $50,000 a year (1), or if they make less than or equal to that number (0).

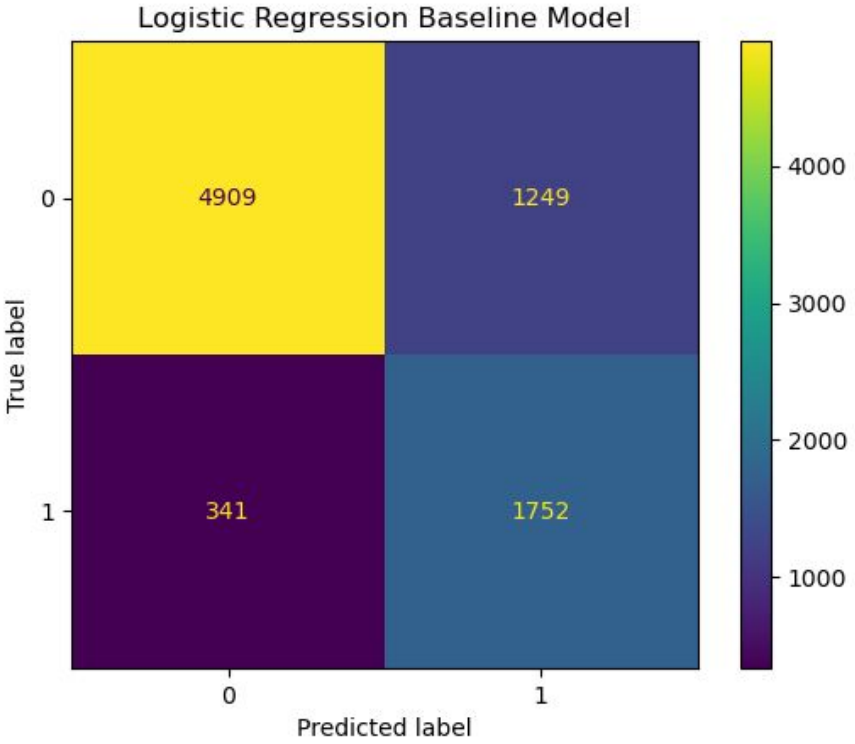
	age	workclass	education	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	income
0	50	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	0	0	0	13	0
1	38	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	0	0	0	40	0
2	53	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	0	0	0	40	0
3	37	Private	Masters	Married-civ-spouse	Exec-managerial	Wife	White	1	0	0	40	0
4	52	Self-emp-not-inc	HS-grad	Married-civ-spouse	Exec-managerial	Husband	White	0	0	0	45	1
...	...	...	...	...	...	...	...	...	...	...	...	...
27498	27	Private	Assoc-acdm	Married-civ-spouse	Tech-support	Wife	White	1	0	0	38	0
27499	40	Private	HS-grad	Married-civ-spouse	Machine-op-inspct	Husband	White	0	0	0	40	1
27500	58	Private	HS-grad	Widowed	Adm-clerical	Unmarried	White	1	0	0	40	0
27501	22	Private	HS-grad	Never-married	Adm-clerical	Own-child	White	0	0	0	20	0
27502	52	Self-emp-inc	HS-grad	Married-civ-spouse	Exec-managerial	Wife	White	1	15024	0	40	1

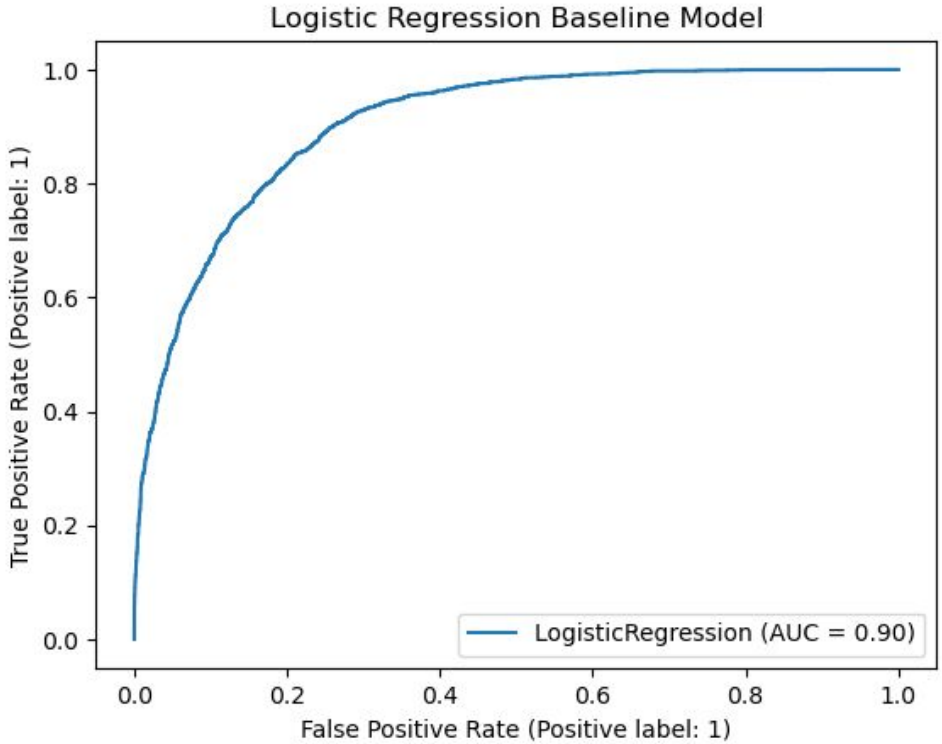
```
y_train.value_counts(normalize = True)
```

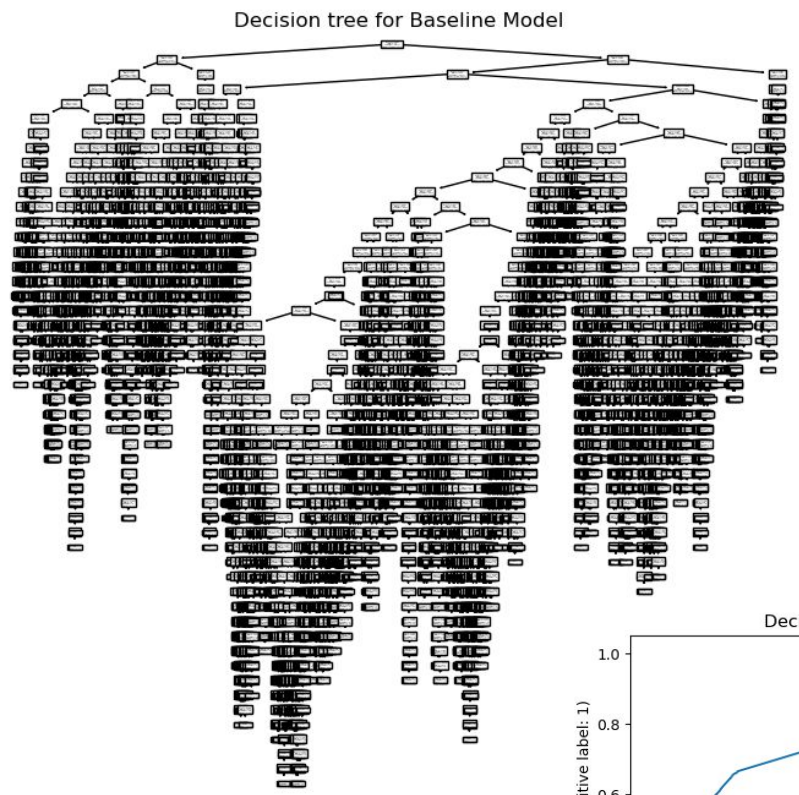
```
income
0    0.745377
1    0.254623
Name: proportion, dtype: float64
```

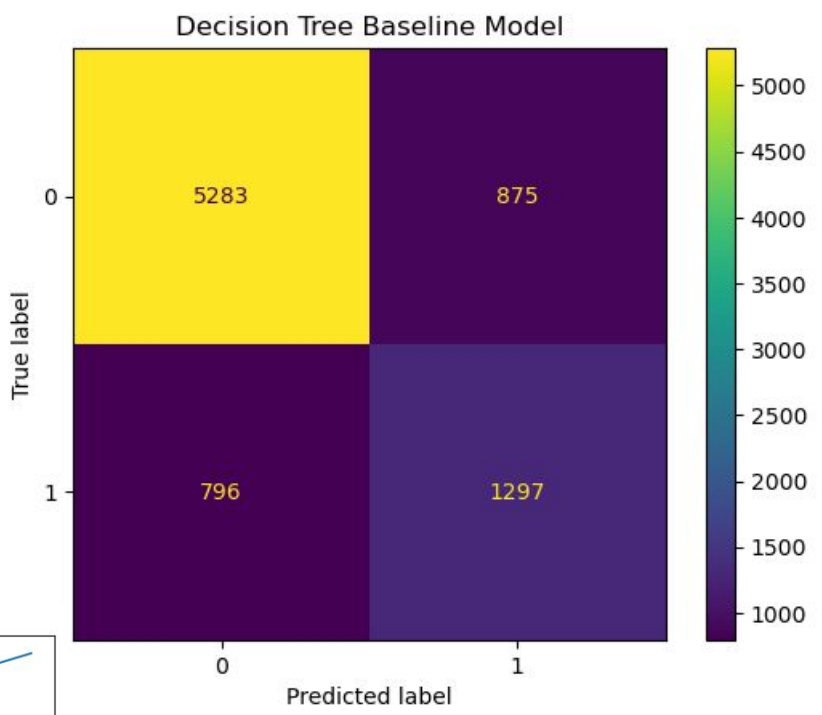
```
y_train_SMOTE.value_counts(normalize = True)
```

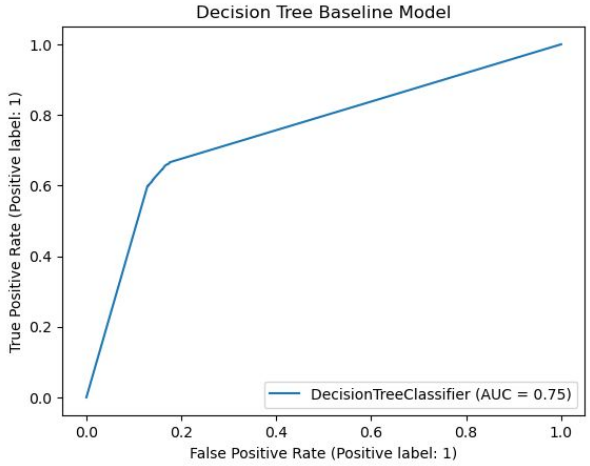
```
income
1    0.5
0    0.5
Name: proportion, dtype: float64
```


Logistic Regression Baseline Model
4909
1249
341
1752
0
1
True label
0
1
Predicted label
4000
3000
2000
1000


Logistic Regression Baseline Model
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
LogisticRegression (AUC = 0.90)


Decision tree for Baseline Model


Decision Tree Baseline Model
5283
875
796
1297
0
1
True label
Predicted label
5000
4500
4000
3500
3000
2500
2000
1500
1000


Decision Tree Baseline Model
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
DecisionTreeClassifier (AUC = 0.75)
0.0
0.2
0.4
0.6
0.8
1.0

Best Performing Models:

	Decision Tree	Logistic Regression
Accuracy train	0.833	0.828
Accuracy test	0.777	0.807
Recall train	0.910	0.859
Recall test	0.852	0.839
F1 train	0.845	0.833
F1 test	0.659	0.689
CV results	0.829	0.827
Precision train	0.809	0.809
Precision test	0.584	0.584

```
# Observing the best hyperparameters for logistic regression

validator.top_hp_lr
```

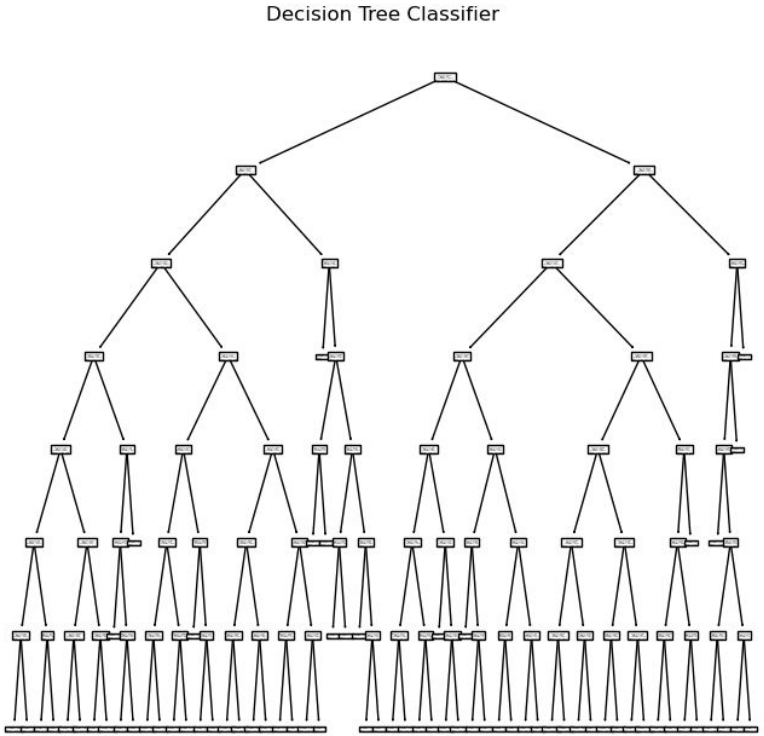
```
[({'C': 1.0,
   'max_iter': 1000,
   'penalty': 'l1',
   'random_state': 42,
   'solver': 'liblinear',
   'tol': 0.001},
  0.8271080139372822,
  0.8074172827536056)]
```

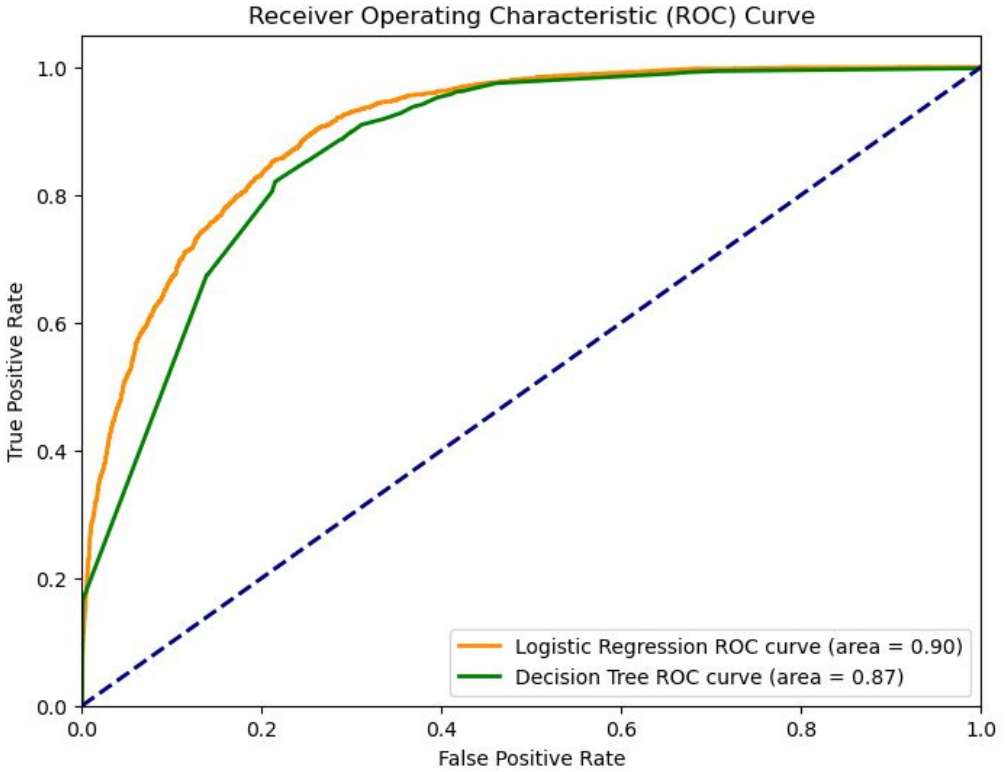
```
#Observing the best hyperparameters for decision trees

validator.top_hp_dt
```

```
[({'criterion': 'gini',
   'max_depth': 7,
   'min_samples_leaf': 2,
   'min_samples_split': 2,
   'random_state': 42},
  0.8286759581881533,
  0.7765119379469155)]
```

Improved Decision Tree:





Recommendations:

- Prefer Logistic Regression: Among the two models, the logistic regression model outperforms the decision tree model, showing better accuracy and precision. It is advisable to prioritize using the logistic regression model for predictions.

- Further Model Improvement: While we have made progress in optimizing these models, it's worth exploring other classification algorithms like K-Nearest Neighbors (KNN) or Random Forest to potentially achieve better performance. These models may offer different insights and capabilities for your specific use case.

Next Steps:

- Cloud Deployment: I would consider deploying the machine learning model to a cloud service, such as AWS or Saturn Cloud. This will enable me to conduct more extensive hyperparameter tuning and make the model accessible for real-world predictions.
- Explore Additional Models: I'd experiment with different classification models, such as K-Nearest Neighbors (KNN), Random Forest, Support Vector Machines (SVM), or Neural Networks. Each model has its strengths and may provide enhanced predictive capabilities.
- Polish Model Pipelines: Streamlining the model training and testing processes by refining the ModelValidator class and pipelines would be essential. Creating efficient and versatile pipelines can help save time and maintain consistency in the modeling workflow.
- Data Enhancement: Importing more up-to-date and detailed data to enhance the quality of predictions would be a priority. Cleaning and maintaining a rich dataset is crucial for accurate modeling.
- Comparative Analysis: Comparing the performance of different models across various genres, directors, or other relevant factors can offer valuable insights into the preferences and trends in the industry.
- By following these next steps and continuously refining the models, I can provide more accurate and insightful predictions, making a valuable contribution to Microsoft's HR practices and further showcasing my data science capabilities.

Contact Me:

Franko Ndou

GitHub: https://github.com/fndou05

LinkedIn: https://www.linkedin.com/in/frankondou/

Email: frankondou@gmail.com