COMP309 A4

Judah Dabora 300476247

Part 1: Regression Performance Metrics

Analysis shows that there is some preprocessing to be done. The pandas isna.any() function does not reveal any missing values. The first column of the data is an index column and can be dropped, as the data frame keeps track of the index.

Regression algorithms perform better on standardized data. Standardizing the data also reduces the variance of output between algorithms, making them easier to compare.

All numerical features of the dataset were scaled with StandardScaler, and the categorical features encoded using onehot encoding.

Regression Accuracy Table

Algorithm	**MSE**	**RMSE**	**RSE**	**MAE**	**Performance Time (s)**
{Linear Regression}	1324177	1150.73	0.08	743.19	0.02
{KNeighbours}	660521.1	812.72	0.04	415.94	1.66
{Ridge Regression}	1324230	1150.75	0.08	743.18	0
{Decision Tree}	532345.7	729.62	0.03	353.47	0.02
{Random Forest}	302534.7	550.03	0.02	273.11	0.72
{Gradient Boosting}	554737.4	744.81	0.03	406.42	0.05
{SGD Regressor}	1329333	1152.97	0.08	742	0
{SVR}	8313686	2883.35	0.51	1373.55	158.7
{Linear SVR}	2455484	1567	0.15	819.6	0
{MLP}	533488	730.4	0.03	373.7	0.02

From the above analysis, we can see that index 7 (Support Vector Regression) had the highest errors and took the longest. It was the worst in every category, Random Forest Regressor had the lowest errors, being the best in every category except performance time, but was third-slowest, and Ridge Regression was the fastest but less accurate than Random Forest.

Random forest was the best for several reasons:

- It requires minimal hyperparameter adjustment, and hyperparameters do not affect its performance much. Where other regressors would have been more effective if tuned, random forest has consistent effectiveness even with default settings.
- It is less computationally intensive than SVR and the geometric KNeighbours algorithm, and so was not the slowest of the processors.
- It handles non-linearity well, because the tree structure of its nested decision trees naturally fits non-linear, polynomial and especially exponential relationships.

Part 2: Classifier Performance Metrics

The data needs to be cleaned as it has several issues:

- The target variable in the training and test sets has leading whitespace. The final character in the test set is a full stop, which must be cleared with a regex as it prevents matches between training and test set.
- The numerical columns must be scaled to improve classifier performance, especially as hyperparameters are being left to default – supplying classifiers with standardized data will decrease the variance of the output.
- The categorical columns are to be onehot encoded so that classifiers can work with the categories numerically, which is required for classifiers like logistic regression.
- After encoding, it is found that the test set lacks a column present in the training set (for the country Holland-Netherlands). This column is added to the correct place in the test set and filled with zeroes to represent that none of the testing instances contain this category.
- The positive class (income) is encoded with a 1 for the class ">50K" and a 0 otherwise. This is necessary as computing the AUC score for some classifiers assumes a class of 1 for the positive class.
- This preprocessing results in dimensionality expansion (from 15 to 108 columns and tens of thousands of rows).

Classifiers Accuracy Table

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
K-Neighbours Classifier	0.83	0.67	0.6	0.63	0.86
Gaussian Naive Bayes	0.55	0.34	0.94	0.5	0.76
SVC	0.86	0.76	0.59	0.67	0.70
Decision Tree Classifier	0.81	0.6	0.61	0.6	0.74
Random Forest Classifier	0.85	0.71	0.59	0.65	0.9
Ada Boost Classifier	0.86	0.74	0.6	0.66	0.91
Gradient Boosting Classifier	0.87	0.79	0.6	0.68	0.92
Linear Discriminant Analysis	0.84	0.72	0.56	0.63	0.89
MLPClassifier	0.84	0.67	0.61	0.64	0.89
Logistic Regression	0.85	0.73	0.6	0.66	0.91

From the above, we can see that Gradient Boosting had the highest accuracy, precision, F1-Score and AUC. Gaussian Naive Bayes had the highest Recall.

Meanwhile, GNB had the lowest accuracy, precision, and F1-Score. Linear Discriminant Analysis had the lowest Recall. Decision tree classifier had the lowest AUC.

Gradient boosting was the most effective classifier because it combines strong points of other classifiers - using multiple decision trees, regression, and a feature score.

As a boosting algorithm, it performed multiple passes and corrected the mistakes of previous algorithms, minimizing the loss functions. Its tree structure is better at handling high dimensional, nonlinear data, and it is robust against hyperparameter tuning. These all contributed to it being the most successful *for the default parameters*.

SVC was least effective because of several likely reasons:

- It assumes data linearity and tries to partition the data using a hyperplane. Gradient boosting handles higher-dimensional data better with its tree structure which naturally accommodates non-linearity.
- SVC is highly sensitive to the parameters passed, and by adjusting the parameters could likely be made far more effective on the given data set. Gradient boosting is often less sensitive to hyperparameter tuning.

Part 3: Optimization

See MyAutoencoder.ipynb