COMP309 – Assignment 4 Report
greenthom – 300536064

Part 1: Performance Metrics in Regression

Question One:

Based on early initial data analysis of diamonds.csv can observe that the shape of the data is 10 columns of 53940 rows. The 10 columns in the dataset are based on diamond qualities that are observed when looking at a diamond. The 10 columns highlighted are carat, cut, color, clarity, depth, table, x, y, z, and price. Carat measures the weight of the diamond, cut refers to how well the diamond has been shaped and faceted, color is graded based on how colorless the diamond is (e.g. D, Z), clarity measures the presence of internal flaws or blemishes, depth is a percentage of the height of the diamond by its diameter, table refers to the width of the flat top facet about the diamonds overall diameter, x, y, and z each refer to the width, length and depth respectively and price is the market value of that specific diamond.

From here I went and checked if there were any null/missing values in the dataset, where I found that there was no missing data across the 10 columns. Then printed out a count of all data value counts per column to get an understanding of the distribution and count of each data value per column. I observed from this that there is a very low count of high-priced diamonds in the dataset as well as a very low amount of high-carat diamonds. Used the .describe() python tool to get an understanding of the min, max, lower and upper quartile. Can notice that all column data mostly fall within the lower and upper quartile range where it is very clustered together, we can also notice that the minimum is close to the lower quartile in most cases excluding the x, y, and z. However, we can also notice that the relation between data around the upper quartile and the max value in the dataset has a big jump in some columns, specifically the carat and table, showing us that these columns have possible outliers. From here I went on to identify the categorical(cut, color, clarity) and numerical(carat, depth, table, x, y, z, price) columns inside of the dataset. From here I went and split the categorical features into ordinal and nominal features, as a list. Cut and clarity were deemed as ordinal features, as cut had a grading system for the diamonds (e.g. very good, ideal, premium) and clarity had a grading system for each clarity level (e.g. SI1, VS2). Color had no natural ordering so I deemed it as a nominal feature and the rest of the numerical columns were encoded using min max scaler.

From here I went and performed my exploratory data analysis (following the stages lined out in the brief), where I first created a histogram of all the processed features in the dataset. From this graph we can observe that based on the carat graph is skewed to the right and most of the diamonds in the dataset appear to have a low carat weight which is clustered around small values. Cut (e.g. very good, ideal) has distinct peaks at certain values but can get no real analysis from this, except that the cut values would correspond to the most common cut grades in the dataset, which we can observe from the value counts as Ideal(3). Clarity also has distinct points at certain points in the dataset. Can observe that the depth percentage distribution is relatively narrow, with most diamonds having a depth around a particular range, indicating consistency in diamond proportions in the dataset. The table percentage also shows a narrow distribution with most diamonds falling within a small range of values suggesting that the table size is fairly consistent for the majority of diamonds.

Plotted a correlation matrix heatmap representing the correlation coefficients between different features of your diamond datasets. Can observe that carat has a strong positive correlation with x (0.98), y(0.95), z(0.95), and price(0.92). This suggests that larger diamonds (higher carat weight) generally have larger

dimensions (x, y, z) and a higher price. Similarly, we can observe that x, y, and z are highly correlated with each other (around 0.95-0.97), as expected because they all represent the physical size of the diamond. The strong positive correlation between carat and price (0.92) indicates that carat is a major factor in determining the price of a diamond. Can also observe weaker correlations on the heatmap, where cut shows a very low correlation with most other features, indicating that cut quality doesn't have a strong linear relationship with the diamond's size (carat) or price. Can also observe that clarity has a weak negative correlation with price (-0.072), meaning that diamonds with higher clarity are not significantly more expensive. Can also observe that depth has almost no correlation with price (-0.011) showing that the depth percentage does not strongly affect the diamond's price. For color correlations we can observe the various color grades (e.g. D, E, F) generally have low correlations with the other features. This implies that color may not have a significant linear relationship with carat, clarity, or price in this dataset. The negative correlations between color grades and features like carat or price (e.g. color d and price at -0.072) suggest that lower-color grades might be associated with slightly larger and more expensive diamonds but the relationships are weak.

After this, I then separate the processed data by specifying the target variable 'income' and the training data (without 'income). Then the data was into train and test sets (x, y) with a test size of 0.3 and a random state of 309. From here I train and predict the regression models.

Question Two:

	score	mse	rmse	rse	mae	extime
lr	0.89	0.01	0.07	9.31	0.05	0.00
knr	0.96	0.00	0.04	9.31	0.05	0.92
rr	0.89	0.01	0.07	9.32	0.05	0.00
dtr	0.97	0.00	0.04	5.10	0.02	0.01
rfr	0.98	0.00	0.03	3.83	0.01	0.56
gbr	0.97	0.00	0.04	5.10	0.02	0.04
sgdr	0.85	0.01	0.08	10.78	0.06	0.00
svr	0.91	0.00	0.07	8.49	0.05	6.96
lsvr	0.86	0.01	0.08	10.24	0.04	0.00
mlpr	0.96	0.00	0.04	5.51	0.03	0.02

Question Three:

- Linear Regression Model got a score of 0.89 which shows that the model did find some relationships. MSE of 0.01 which is a low error rate. RMSE of 0.07 shows low residuals. RSE of 9.31, suggesting the model is capturing around 90% of the variance. MAE of 0.05, pointing to a small average error. Execution time of 0.00 (2dp) making it very efficient in terms of computation time.
- K-Nearest Neighbors Regressor got a score of 0.96 showing that the model found strong relationships and was fitted well. Got an MSE of 0.00 which is one of the lowest among all algorithms. Got an RMSE of 0.04 which is very low. RSE of 9.31 and MAE of 0.05, both similar to Linear Regression. Got an execution time of 0.92 which is a relatively high computation time compared to other algorithms but still performs well.
- Ridge Regression got a score of 0.89 which is identical to linear regression. Has an MSE and RMSE score of 0.01 and 0.07 respectively which is very similar to linear regression. Has an RSE score of

9.32 which is slightly higher than linear regression. Has an MAE of 0.05 which is the same as LR and KNR. Has an execution time of 0.00 (2dp) which is very efficient.

- Decision Tree Regressor has a score of 0.97, which is one of the highest scores, showing great predictive power. Has an MSE of 0.00 and an RMSE score of 0.04 which has very low residuals. Has a RSE score of 5.10, which is the second lowest, meaning it captures a lot of variance. Has an MAE score of 0.02 which is the lowest MAE score, suggesting very accurate predictions. Has an execution time of 0.01 which is efficient.
- Random Forest Regressor has a score of 0.98 which is the highest score, indicating good predictive accuracy. Has an MSE of 0.00 similar to DTR and KNR. Has an RMSE of 0.03, which is the lowest residual. Has a RSE of 3.83, which is the lowest, showing that it captures almost all the variances. Also has a MAE of 0.01 which is the lowest average error. Has an execution time of 0.56 which is a moderate computation time.
- Gradient Boosting Regressor has a score of 0.97 which is a very high predictive accuracy. Has an MSE and RMSE of 0.00 and 0.04 respectively, indicating a very low error and residuals. Has a RSE of 5.1 similar to DTR which was also very effective. Has an MAE of 0.02 indicating good precision. Has an execution time of 0.04 which is very efficient.
- Stochastic Gradient Descent Regressor has a score of 0.85, the lowest of all the models. Has an MSE and RMSE of 0.01 and 0.08 respectively, indicating higher error and residuals compared to other models. RSE of 10.78, which is the highest residual sum showing it captures the least variance. MAE of 0.06, the highest average error. Has an execution time of 0.00 showing that it is efficient but not very good performance-wise.
- Support Vector Regressor has a score of 0.91 which is a fairly high accuracy. Has an MSE and RMSE of 0.00 and 0.07 respectively, indicating low error and residuals. Has a RSE of 8.49 capturing most of the variance. Has a MAE of 0.05 which is an average error compared to other models. Has an execution time of 6.96, the highest, requiring a lot of computational time.
- Linear Support Vector Regressor has a score of 0.86 which is lower than SVR and many other models. Has an MSE and RMSE of 0.01 and 0.07 respectively which is slightly higher than SVR in terms of error and residuals. Has an RSE of 10.24, capturing less variance than other models. Has an MAE of 0.04 which is lower than SVR. Has an execution time of 0.04 which is very efficient but not the best-performing model.
- Multi-layer Perceptron Regressor has a score of 0.96 which is a high predictive accuracy. Has an MSE and RMSE of 0.00 and 0.04 respectively, indicating a very low error and residuals. Has an RSE of 5.51 capturing a lot of variance. Has a MAE of 0.03 which is a very low average error. Has an execution time of 0.02 which is very efficient in time and performance.

The Random Forest Regression stands out as the best performer across most metrics, with the highest score (0.98), lowest RSE (3.83), and MAE (0.01). It captures almost all the variance and makes extremely accurate predictions with low residuals. However, it has a moderate execution time of 0.56. In terms of execution time, Linear Regression, Ridge Regression, and SGD Regression are the fastest models, taking almost no time to compute. However, their performance is not the best compared to other models, especially for SGD Regression, which has the lowest score and highest error metrics. K-Neighbors Regression and Multi-Layer Perceptron Regression offer great predictive accuracy (0.96) with low error metrics, though they come with higher execution times (0.92 and 0.02 respectively). Models like Decision Tree Regression, Gradient Boosting Regression, and Multi-Layer Perceptron Regression also perform exceptionally well in terms of accuracy, with scores around 0.97 and low MSE/RMSE values and minimal average error. SGD Regression and Linear SVR perform poorly in comparison to the other models, as they show higher errors and capture less variance, as indicated by their higher RSE values and lower scores.

Part 2: Performance Metrics in Classification

Question One:

On initial data analysis, we can observe that the adult.data set has no column names to indicate what each column data represents for the dataset. Through online investigation of where these datasets originated, we can see that as per the UC Irvine Machine Learning Repository, the dataset was originated to predict whether the annual income of an individual exceeds $50K/yr based on census data. Can also see that it has both categorical and integer types. By using this we can create a list of column names and assign these to when the dataset is read. Can observe through the use of .shape that it has 32561 instances and 15 features in total.

Printed out the column value counts to understand each value in the columns. Through this, we can observe that there is missing data in the work class column indicated by '?', missing data in the occupation column indicated by '?', and missing data in the native country indicated by '?'. From there I applied for the '?' values in each column to be replaced with NaN values. From here I printed out the total amount of missing/null values in each column so I could understand how many values of ?/NaN were in the working class, occupation, and native-country columns. I did this so I could determine if it wasn't necessary to impute or just drop the columns entirely. Can observe that there is a total of 1836 missing values in the work class column and 1843 missing values in the occupation column. Due to the amount of missing data in these two columns, it is easier to impute so when training the classification models, it can get a better understanding of the dataset. There are 583 total missing values in native-country however due to printing out the value counts before we can see that this census/data is mostly made up of United States countries.

From here I separate the dataset into categorical(education, marital status, workclass, occupation, relationship, race, sex, native-country, income) and numerical(age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week) features to perform imputation. Used SimpleImputer with 'most_frequent' as the strategy for categorical and 'median' as the strategy for numerical. Fitted and transformed the datasets based on this. Then went and checked that it had correctly performed imputation by checking for missing/null values. From observing value counts we can a couple of things based on the columns. The age column can show us that the most common ages are 36, 31, 34, 23, and 35 with around 870-900 instances for each, indicating a younger, working-age population in the dataset. The lower count of individuals above 80 (e.g. 83, 88) suggests that there are very few elderly people in/participated in the dataset/census. For the working class, most individuals (24,532) belong to the private sector, followed by "self-emp-not-inc" (2541) and then government sectors (local, state, and federal). There a very few individuals under without-pay and never-worked, indicating the dataset predominantly includes employed individuals. The education column, "HS-grad" is the most frequent education level (10, 501), followed by "Some-collect" and "Bachelors". The dataset contains very few individuals with lower education levels, like "1st-4th grade" (168) and "Preschool" (51) which can be expected from understanding the census purpose. Education-num column is a numerical feature part of the education column showing the same distribution. Marital-status is the most frequent marital status is Married-civ-spouse (14976), followed by Never-married (10,683). The dataset has very few people categorized as Married-AF-spouse (23, reflecting the rarity of this category. The relationship column has most of its features labeled as husband(13,194) or not in family (8305). The wife category is much smaller (1568) indicating that more men (or husbands) are represented in the dataset. This can be backed up in the sex column as well where there are more males (21,690) than females (10,771) in the dataset, indicating a gender imbalance. For hours-per-week the majority of people work between 40-50 hours per week

indicating that most individuals work full-time. Fewer individuals are working extreme hours over 80 which suggests that such cases are rare in this dataset.

Plotted a histogram with KDE (Kernel Density Estimation) to get a visual representation of the distribution of various features in the dataset. Can observe from the age column that the distribution shows that the majority of the population is between 20 and 50 years old. The peak is around the 30-40 range which suggests this is the most common working age in the dataset, and the density decreases as age increases. The education graph shows the numerical counterpart of the education variable and follows the same pattern, with distinct peaks corresponding to common education levels like high school (9), some college (10), and bachelors (13). From observing the histogram, I can notice that the dataset is dominated by U.S. residents who work in the private sector, with a significant proportion being white males. There are fewer individuals from minority racial groups and other countries. The data also shows that most individuals, earn <=50K and report no capital gains or losses, indicating a lower-income group. Most people in the dataset also work a standard 40-hour week, and the dominant occupations are in professional specialties or managerial roles.

Created a boxplot that showed the distribution of dataset categories in comparison to income categories (<=50K and >50K). Observations obtained from this are that individuals in the self-emp-inc and federal-gov work classes are more represented in the >50K group, indicating that self-employed incorporated individuals and federal government employees are more likely to earn higher incomes. The without-pay and never-worked categories only appear in the <=50K group, as expected. In the education-num graph, similar to the education boxplot, the higher education number (which corresponds to more years of education is associated with higher income levels. Individuals with higher education-num values (10-16) are more likely to be in the >50K group. The <=50K group has a wider range of education levels, while the >50K group is more concentrated around higher values. From earlier ID and ED analysis, we observed that most of the dataset is made up of and is dominated by U.S residents (printed out a solo bar graph to visualize this, to make this clear), with 24720 of the data being made up of U.S citizens and only 7841 being other countries. Of these 7841 countries the third highest count (found from early IDA) of 583 following Mexico which had 643, were missing values that were imputed. Due to the lack of non-U.S data, which would result in no association of relationships found, I changed all data types in the native country that were non-US to ‘other’ so it would be US vs non-U.S. This would allow for clearer relationships to be understood.

Created a correlation matrix heatmap that showed the linear relationships between some of the numerical features in the dataset. Only analyzed the age, fnlwgt, education, education-num, capital-gain, capital loss, hours-per-week, and income in the heatmap. From the heatmap, we can make some observations. The correlation between age and income is 0.23, which is a positive but weak correlation. This suggests that older individuals are more likely to have higher incomes, but age alone is not a strong predictor. The correlation between education and education-num is 0.36, which makes sense because these two variables are highly related (education-num represents the number of years of education, which correlates with education levels). The correlation between education-num and income is 0.34, which shows that higher education levels are moderately associated with higher income. The correlation between capital gain and income is 0.22, indicating that individuals with capital gains are more likely to earn higher incomes. The correlation between hours-per-week and income is 0.23, suggesting that working more hours per week is associated with higher income, although this is not a very strong correlation. The fnlwgt variable shows very weak correlations with most other features, including income (-0.0095). This suggests that fnlwgt is not strongly related to income or the other numerical features in the dataset. The correlation between capital loss and income is 0.15, indicating a weak positive correlation. Individuals reporting capital losses tend to have slightly higher incomes, but the relationship is not

strong. Age shows small positive correlations with income (0.23), capital gain (0.078), and hours-per-week (0.069), but none of these correlations are particularly strong.

Further improvement in removing/adjusting the dataset involved the removal of the education column due to the education-num columns representing the same data/values just in categorical and numerical form. By adjusting this the risk of overfitting, the dataset would lessen and allow more relationships to be found when being trained on the models.

For processing, I split the data into ordinal (education-num), nominal (class, occupation, relationship, race, sex, native-country, marital status), and numerical (age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week, income) features. From here I fit and transformed the numerical features using MinMaxScaler with no parameters, fit and transformed ordinal features using OrdinalEncoder, and fit and transformed nominal features using OneHotEncoder. From here I separated my target column ('income') and created a test set and removed the target column from my train set. Then performed making train and test sets (x, y) to train my model. Performed the same processing steps when reading and processing the adult.test set.

Question Two (Report the results (keep 2 decimals) of all the 10 classification algorithms on the given test data in terms of classification accuracy, precision, recall, F1-score, and AUC. You should report them in a table:

When training and testing the adult.data:

	accuracy	precision	recall	f1	auc	extime
knn	0.83	0.66	0.58	0.62	0.74	0.58
gnb	0.61	0.38	0.95	0.54	0.73	0.00
svm	0.85	0.75	0.58	0.65	0.76	0.00
dt	0.82	0.62	0.63	0.63	0.76	0.00
rf	0.86	0.75	0.62	0.68	0.78	0.11
ada	0.86	0.77	0.60	0.68	0.77	0.03
gbc	0.87	0.80	0.59	0.68	0.77	0.01
lda	0.84	0.71	0.57	0.63	0.75	0.00
mlpc	0.85	0.75	0.59	0.66	0.76	0.01
lrc	0.85	0.74	0.58	0.65	0.76	0.00

When testing from the adult.test data on the fitted models:

	accuracy	precision	recall	f1	auc	extime
knn	0.86	0.74	0.65	0.69	0.79	1.41
gnb	0.61	0.38	0.95	0.54	0.73	0.01
svm	0.85	0.74	0.58	0.65	0.76	0.00
dt	0.95	0.88	0.89	0.89	0.93	0.00
rf	0.96	0.94	0.89	0.91	0.93	0.35
ada	0.86	0.77	0.61	0.68	0.77	0.10
gbc	0.87	0.80	0.60	0.69	0.78	0.03
lda	0.84	0.71	0.57	0.63	0.75	0.00
mlpc	0.86	0.77	0.61	0.68	0.78	0.03
lrc	0.85	0.73	0.59	0.65	0.76	0.00

Question Three: (Find the two best algorithms according to each of the performance metrics. Are they the same? Explain why?)

Based on analyzing the performance metrics of accuracy, precision, recall, f1-score, AUC, and execution time across both tables we can make some conclusions. For accuracy on the training data, we can see that Gradient Boosting Classifier (GBC) had the best accuracy with 0.87, with AdaBoost (ADA) having the second-best accuracy of 0.86. For accuracy on the test data, we can see that Random Forest (RF) had the best accuracy with 0.96, followed closely by Decision Tree (DT) with an accuracy of 0.95. For precision on the training data, we can see that the Gradient Boosting Classifier (GCB) had the best precision of 0.8, followed by AdaBoost (ADA) with a precision of 0.77. For precision on the testing data, we can see that the highest precision is Random Forest (RF) with a precision of 0.94, followed by Decision Tree with a precision of 0.88. For recall on the training data, we can see that Gaussian Naive Bayes (GNB) has the highest recall of 0.95, followed by Decision Tree (DT) with a recall of 0.63. For recall on the testing data, Random Forest (RF) and Decision Tree (DT) had the highest recall of 0.89. For the F1-Score on the training data, the highest F1 score is AdaBoost (ADA), Random Forest (RF), and Gradient Boosting Classifier with a score of 0.68. For F1-Score on the testing data, the highest F1 score is Random Forest (RF) with a score of 0.91 followed by Decision Tree (DT) with a score of 0.89. For AUC on the training data, the best AUC is Random Forest (RF) with an AUC score of 0.78, followed by Gradient Boosting Classifier (GBC) and AdaBoost (ADA) with a score of 0.77. For AUC on the testing data, the best AUC is Random Forest (RF) and Decision Tree (DT) with a score of 0.93. For Execution Time on the training data, GNB, SVM, DT, LDA, and LRC all got good execution times of 0.00. It is hard to make a good analysis of that due to the execution times being rounded to 2d.p. However, from earlier analysis before converting these times the slowest times were LRC and SVM. For Execution Time on the testing data, there was a similar situation as per the training data, with SVM, DT, LDA, and LRC scoring 0.00.

In both training and test data, Random Forest (RF) and Decision Tree (DT) consistently outperform other models in key performance metrics like accuracy, precision, recall, F1-score, and AUC, making them the best overall performers for this dataset. While Gradient Boosting Classifier (GBC) and AdaBoost (ADA) perform well in training, their performance drops on the test data, indicating potential overfitting. The differences between the training and test results indicate that GBC and ADA might be overfitting to the training data, capturing noise that doesn't generalize well to unseen data. In contrast, ensemble models like Random Forest and Decision Tree are better at generalizing and handling unseen data, which is why they consistently perform well across both sets. If execution time is a major concern, Logistic Regression

and SVM provide the fastest results, though their predictive power is not as strong as the ensemble methods. In conclusion, Random Forest and Decision Tree are the best algorithms across most metrics when considering generalization to unseen data, making them more reliable choices for classification in this scenario.

References

- ChatGPT (help with bug fixing on models in classification, as well as help on understanding part 3 implementation (understanding Torch and SGD)
- https://scikit-learn.org/stable/modules/model_evaluation.html (understand metric implementation for part 2 for the models)
- https://scikit-learn.org/stable/api/sklearn.svm.html (understand SVM implementation for regression models)
- https://seaborn.pydata.org (understanding seaborn implementation for EDA and IDA on both regression and classification models)