COMP309

Assignment 4

Elliott Rose - 300540768

Part 1)

Using an initial data analysis (IDA) I was able to find the following information about the diamonds.csv data set:

- Contains 53,940 instances.
- The X, Y and Z values in the data have minimums of 0. This means there are some missing values, since doesn't make sense for the dimensions to be 0.
- There is an unnamed column which shows the counts the instance, this information isn't valuable so can be removed.
- 3 categorical features, namely: Cut, color, and clarity.
- 7 numerical features (excluding "Unnamed"), those being: Carat, depth, table, x, y, z, price.

Due to these findings the following steps in my preprocessing were taken:

- The first column was removed as it has no relevance to the data/ model prediction.
- Imputed the 0 values in the dimension columns (x, y and z) with the mean.
- Encoded categorical features using an OrdinalEncoder because values in the categorical features have a clear order which would influence the diamond price.
- Split the target feature from the rest of the data to perform a train test split.
- Scaled numerical features using a StandardScaler as the feature values are all using different scales.

Shown below are the results from the different models:

Model	*MSE*	*RMSE*	*RSE*	*MAE*	*RUNTIME*
LinearRegression()	*2,335,265.55*	*1,528.16*	*579.83*	*895.14*	*0.004603*
KNeighborsRegressor()	*2,207,730.63*	*1,485.84*	*548.17*	*831.48*	*0.185915*
Ridge()	*2,335,259.54*	*1,528.16*	*579.83*	*895.33*	*0.000000*
DecisionTreeRegressor()	*3,613,430.34*	*1,900.90*	*897.19*	*1,050.34*	*0.011345*
RandomForestRegressor()	*1,990,624.58*	*1,410.89*	*494.26*	*792.20*	*0.517433*
GradientBoostingRegressor()	*1,879,375.87*	*1,370.90*	*466.64*	*776.10*	*0.015626*
SGDRegressor()	*2,347,426.68*	*1,532.13*	*582.85*	*883.99*	*0.015625*
SVR()	*7,577,653.59*	*2,752.75*	*1,881.48*	*1,349.92*	*63.247901*
LinearSVR()	*3,301,263.80*	*1,816.94*	*819.68*	*985.26*	*0.000000*
MLPRegressor()	*2,122,725.74*	*1,456.96*	*527.06*	*823.62*	*0.017392*

Based off the results shown above we make the following conclusions and comparisons:

- GradientBoostingRegressor consistently outperformed all other models, achieving the lowest values in each category. Its MSE of 1,879,375.87 and RMSE of 1,370.90 indicate better predictive accuracy, while its RSE of 466.64 and MAE of 776.10 highlight its robustness and ability to generalize well.
- In contrast the SVR model showed the worst performance with the highest MSE (7,577,653.59), RMSE (2,752.75), RSE (1,881.48), and MAE (1,349.92). This suggests that SVR struggled hugely with this dataset. RandomForestRegressor performed well, with the second-best MSE and RMSE, proving to be an effective alternative, though not as optimal as GradientBoosting.
- Meanwhile, the DecisionTreeRegressor and LinearSVR displayed relatively high error rates, hinting at potential issues with overfitting and underfitting. Simple linear models like LinearRegression and Ridge were outperformed by the more complex ensemble methods.
- In conclusion, the GradientBoostingRegressor is the best choice for this dataset due to its superior performance across all metrics. Ensemble models such as GradientBoosting and RandomForest show clear advantages over simpler models like LinearRegression and Ridge. Finally complex models like SVR should be avoided due to their poor performance.

Part 2)

By carrying out an exploratory data analysis (EDA) I was able to find the following information:

- There are some missing values in the data. These are denoted with "?".
- There are "." characters at the end of every value in the income feature.
- There are some categorical features which require encoding.
- Both the data sets have no column names.
- The test data has a rogue line as the first line which is rubbish.
- The education-num column is the ordinally encoded version of the education column.
- Some other columns can be reduced after encoding such as marital status and capital etc.

Due to these findings the following steps in my preprocessing were taken:

- Made a list of column names for the adult data by finding them online. Then set column names of the train and test data to these names.
- Removed the first entry in the test data as it's not an instance.
- Replaced all the "?" (missing values) with np.nan. I didn't impute them because I didn't think it was necessary as it's not a large portion of the data.
- Removed the "." character from the target feature.
- Encoded the categorical features using OneHotEncoder. However, I encoded the native-country feature separately into True (United-States) and False (not United-States) because if I OneHotEncoded that it results in a very large number of features.
- Scaled the numerical features using a StandardScaler as the feature values are all using different scales.
- Removed education column as education-num already contains the relevant education information and by removing it the dimensionality is reduced.
- To further reduce dimensionality, I combined the capital-loss and capital-gain features into net-capital and changed marital status to True (married) and False (not married). Overall, this reduced my dimensions by around 20. Further dimensionality reduction could be done with the work-class and occupation features. However, I chose to leave these as they were as it would be too tedious.

MODEL	ACCURACY	PRECISION	RECALL	F1-SCORE	AUC
LogisticRegression()	0.81	0.67	0.36	0.47	0.65
KNeighborsClassifier()	0.80	0.60	0.44	0.51	0.67
DecisionTreeClassifier()	0.77	0.51	0.53	0.52	0.69
RandomForestClassifier()	0.81	0.62	0.51	0.56	0.71
AdaBoostClassifier()	0.84	0.79	0.41	0.54	0.69
GradientBoostingClassifier()	0.84	0.78	0.45	0.57	0.71
SVC()	0.82	0.77	0.35	0.48	0.66
MLPClassifier()	0.82	0.71	0.43	0.54	0.69
GaussianNB()	0.80	0.75	0.23	0.35	0.60
LinearDiscriminantAnalysis()	0.80	0.66	0.32	0.43	0.63

Based off the results shown above we make the following conclusions and comparisons:

- The two best algorithms for each performance metric vary, highlighting different strengths. GradientBoostingClassifier consistently ranks among the top across several metrics, including Accuracy (0.84), Precision (0.78), F1-Score (0.57), and AUC (0.71). This indicates that it provides a good balance between precision, accuracy, and overall performance. Similarly, AdaBoostClassifier performs well in terms of Accuracy (0.84) and Precision (0.79), but its lower Recall (0.41) shows that while it identifies positives well, it misses many true positive cases.

- On the other hand, RandomForestClassifier excels in Recall (0.51) and F1-Score (0.56), meaning it is better at capturing positives compared to others, though it sacrifices a bit of precision. The DecisionTreeClassifier, despite performing well in Recall (0.53), struggles with other metrics, making it less consistent overall.

- In conclusion, GradientBoostingClassifier and RandomForestClassifier emerge as the most well-rounded algorithms, but they excel in different areas. GradientBoosting is more balanced, while RandomForest is better at capturing positive cases. The best choice depends on whether precision or recall is more important for the specific task.