Project Final Report

Konstantinos Giantsios, AM: 83, giantsik@csd.auth.gr

Task 1

Preprocess

The NSL-KDDTrain set has 125973 entries and contains 41 features and the NSL-KDDTest is composed of 22544 entries and has one more column which is the target column. The datasets don't have any null value on every column, so there is no need for some filling or imputing on the preprocessing phase. The dataset has 3 features ['protocol_type', 'service', 'flag'], that are categorical and one hot encoding is used to transform them to numerical values as the rest features. The last step of the preprocessing phase is to apply the min max scaling method to normalize the dataset's features.

Training

In the training phase, unsupervised methods will be used because there is no target column on the NSL-KDDTrain set. Methods that are used:

- **K-means clustering**: is the most common type of clustering and a base model to start with unsupervised learning. It's a centroid-based algorithm and the simplest unsupervised learning algorithm. This algorithm tries to minimize the variance of data points within a cluster. K-means is computationally expensive but using sklearn implementation didn't cause any problem.
- **BIRCH** (Balance Iterative Reducing and Clustering using Hierarchies): it is believed that this algorithm has better performance on large dataset than K-means algorithm. It breaks the data into little summaries that are clustered instead of the original data points. The summaries hold as much distribution information about the data points as possible. The main downside with this algorithm is that only numerical values must be used which is not a problem with my approach because of data preprocessing.
- **Gaussian Mixture**: this algorithm has better performance than K-means when the data are non circular. The Gaussian mixture model uses multiple Gaussian distributions to fit arbitrarily shaped data.

In the finetuning process for BIRCH model, the value of threshold (the radius of the subcluster obtained by merging a new sample and the closest subcluster should be lesser than the threshold. Otherwise a new subcluster is started. Setting this value to be very low promotes splitting and vice-versa) is finetuned. For the Gaussian Mixture model, the covariance type is finetuned. After the training two clusters are created, the one with the least elements is the positive class ("attack") and the other is the negative class ("normal").

Evaluation

For the evaluation phase Accuracy metric is used to decide the best model

Task1	
method	accuracy
kmeans	0.5292
Birch(threshold = 0.3)	0.6399
Birch(threshold = 0.6)	0.6637
Birch(threshold = 1)	0.6592
GaussianMixture(covariance_type="full")	0.5212
GaussianMixture(covariance_type="tied")	0.7335
GaussianMixture(covariance_type="diag")	0.5786
GaussianMixture(covariance_type="spherical")	0.7355

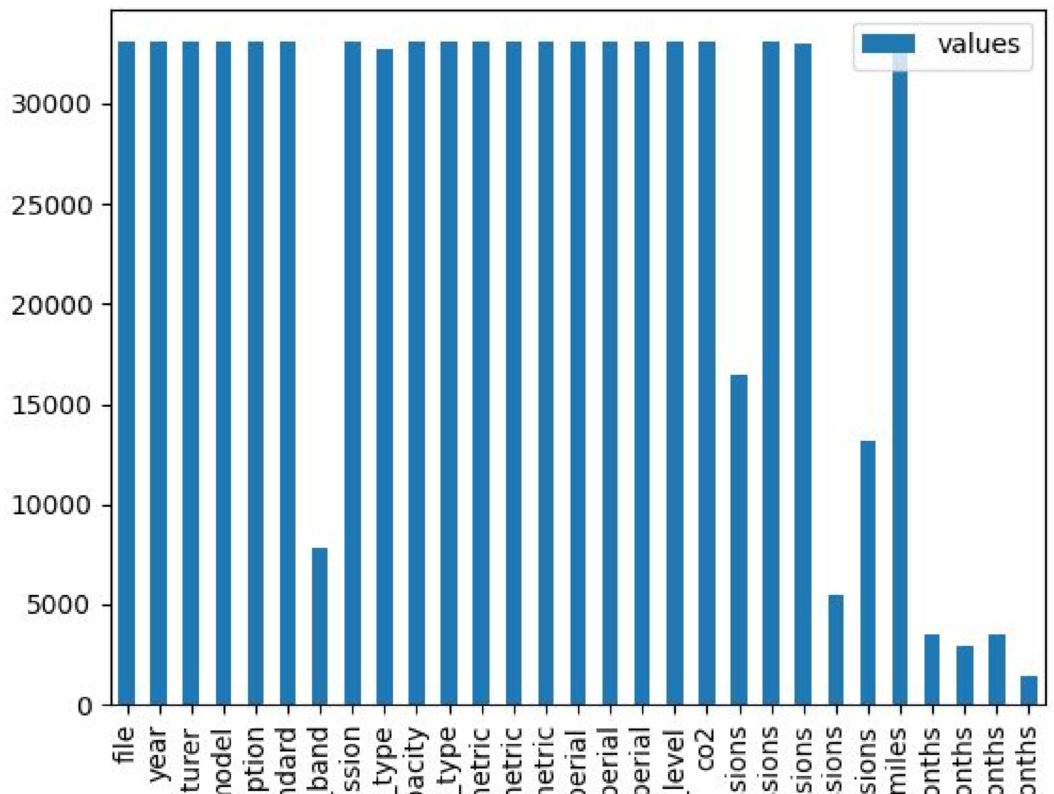
Table 1: Task 1 results

As you can see the best model is GaussianMixture(covariance_type="spherical"), as expected Gaussian Mixture has the best performance because of its ability to detect complex clusters.

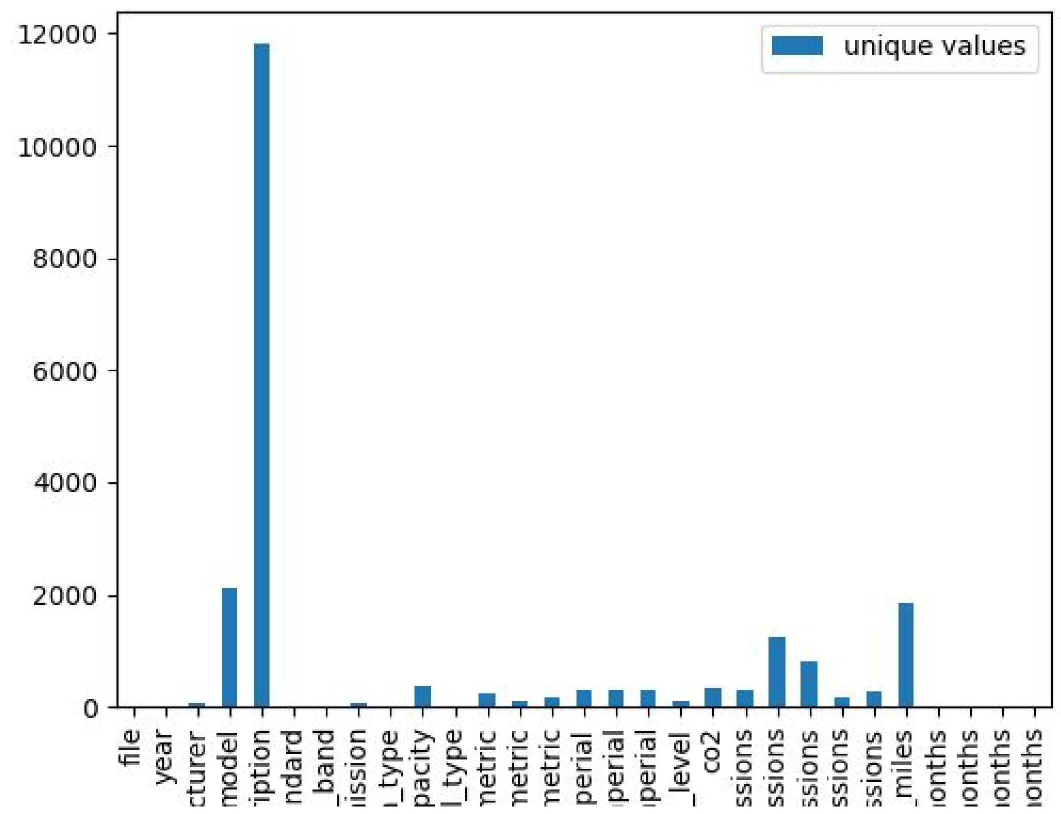
Task 2

Preprocess

The fuel_emissions dataset has 33088 entries and one more column which is the target column. Some examples from the dataset have null values on target columns; these examples are removed, so the dataset has 33078 entries after this transformation. The next step is to check for null values on features.



As you can see from the above histogram some features have a lot of null values so those features whose null values are more than their count of values are removed. Those features are: ["tax_band", "thc_nox_emissions", "particulates_emissions", "standard_12_months", "standard_6_months", "first_year_12_months", "first_year_6_months"]. The next step of the preprocessing phase is to check the number of unique values of the features.



As you can see from the above picture, a feature like description is not useful and is removed because there are a lot of values for this feature and it will not help to find a partner for the regression model. Another feature that is removed is the file because it is not useful on the prediction if we have new examples that are not in a file. After the removal of redundant features a splitting of dataset into train set (80%) and test set (20%) with shuffling is happening. The dataset has 5 features ['manufacturer', 'model', 'transmission', 'transmission_type', 'fuel_type'], that are categorical and one hot encoding is used to transform them to numerical values as the rest features. The next step of the preprocessing phase is to apply a simple imputing method filling null values of feature with the mean of the feature after this min max scaling method is applied to normalize the dataset's features in feature range (-1, 1).

Training

In the training phase, supervised methods for regression will be used. Methods that are used:

- **LinearSVR**: the model produced by Support Vector Regression depends only on a subset of the training data, because the cost function ignores samples whose prediction is close to their target. LinearSVR is an SVR model with a linear kernel and is used because it is a faster implementation than SVR.
- **DecisionTreeRegressor**: Decision tree applied to regression problems.
- **KNeighborsRegressor**: Regression based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated with the nearest neighbors in the training set.
- **RandomForestRegressor**: Random Forest applied to regression problems.

In the finetuning process for LinearSVR model, the value of C (Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.) is finetuned. For the DecisionTreeRegressor model criterion of splitting is finetuned. For the KNeighborsRegressor model the value of k is fintuned. Finally for the RandomForestRegressor model the number of estimators is finetuned.

Evaluation

For the evaluation phase MAE, MAPE and RMSE metrics are used to decide the best model. Sklearn's implementation of MAPE notifies us that they do not represent the output as a percentage in range [0, 100]. Instead, they represent it in range [0, 1/eps].

Task2			
method	MAE	MAPE	RMSE
LinearSVR(C=0.1)	153.5202	1037245000158700.0000	245.9594
LinearSVR(C=1)	120.0163	1059151775304240.0000	176.0679
LinearSVR(C=10)	118.8656	1034993473368990.0000	167.7859
DecisionTreeRegressor(criterion="mse")	5.9739	0.0046	50.6325
DecisionTreeRegressor(criterion="friedman_mse")	5.8527	0.0045	50.2827
DecisionTreeRegressor(criterion="mae")	8.2028	0.0056	65.0335
DecisionTreeRegressor(criterion="poisson")	309.8520	1554715291555750.0000	446.7152
KNeighborsRegressor(k=1)	417.5300	1700544515886400.0000	570.3316
KNeighborsRegressor(k=3)	418.2423	1616342541595740.0000	574.1881
KNeighborsRegressor(k=5)	413.6488	1665186902246000.0000	571.6319
RandomForestRegressor(estimators=100)	7.1658	202308012281516.0000	47.6934

RandomForestRegressor(estimators=200)	7.1116	177560677872046.0000	47.8210
RandomForestRegressor(estimators=1000)	7.1822	204258256074203.0000	48.1444

Table 2: Task 2 results

It is clear that the best model is DecisionTreeRegressor(criterion="friedman_mse"). Furthermore simple regressors are used in experiments and they didn’t have the expected performance so decided to not include them. In general the methods with decision tree algorithm have outperformed the other methods.