EPA Vehicle Emission Score Prediction using Machine Learning

Group 4:
Sayantan Majumdar
Dawit Wolday Asfaw
Cong Shen
Divya Reddy Manku
Akhil Reddy Annreddy

Introduction

- Greenhouse gases (GHG) are responsible for trapping heat, thereby making the planet warmer.

- In the US, the transportation sector generates the largest share of GHG emissions (29% of 2019 GHG emissions) from burning fossil fuels.

- The Environmental Protection Agency (EPA) emissions score (smog rating) reflects vehicle tailpipe emissions (CO_2) that contribute to local and regional air pollution

Problem Description

- Building a machine learning model to predict the EPA emission score from different vehicles operating in the US.
- Classification problem, emission score: 1 (worst) - 10 (best)
- Compare different machine learning models to assess their performance on the data sets
- Broad business objective: Enabling policymakers to address critical issues in the sustainable transportation industry

Data Source and Collection

- Available on the EPA fuel economy portal [https://www.fueleconomy.gov/]
- Two data sets:
  - ➢ Vehicle data (83 variables, 44075 observations)
  - ➢ Emissions data (8 variables, 42442 observations)
- The EPA has generated annual reports, but there are no existing publicly available notebooks having detailed machine learning workflows

https://www.fueleconomy.gov/feg/pdfs/guides/FEG2021.pdf

Data Source and Collection

Data Set	Feature Name	Details
emissions	score	EPA 1-10 smog rating for fuelType1 (target variable)
vehicle	fuelType1	For single fuel vehicles, this will be the only fuel. For dual fuel vehicles, this will be the conventional fuel.
vehicle	highway08	Highway MPG for fuelType1
vehicle	barrels08	Annual petroleum consumption in barrels for fuelType1
vehicle	year	Model year
vehicle	VClass	EPA vehicle size class
vehicle	phevBlended	If True, this vehicle operates on a blend of gasoline and electricity in charge depleting mode

Potential Issues & Challenges

- Data Curation: Merging the Vehicles and Emissions data sets
- Feature Engineering
- Model Complexity: Hyperparameter Tuning
- Evaluation Metrics

Research Questions

- *Which predictors are most suited for predicting the EPA emission score?*
  - Which machine learning model works best for this data set?
  - Which vehicle class has poor emission ratings?

Data Management

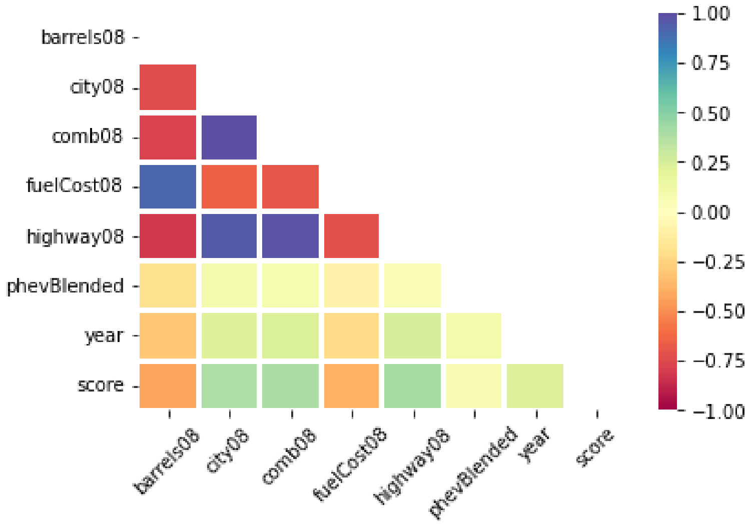
- After merging and cleaning data (removing missing and incorrect values), we have 28556 non-redundant observations.
- Data Transformation: Aggregating fuelType1 variable

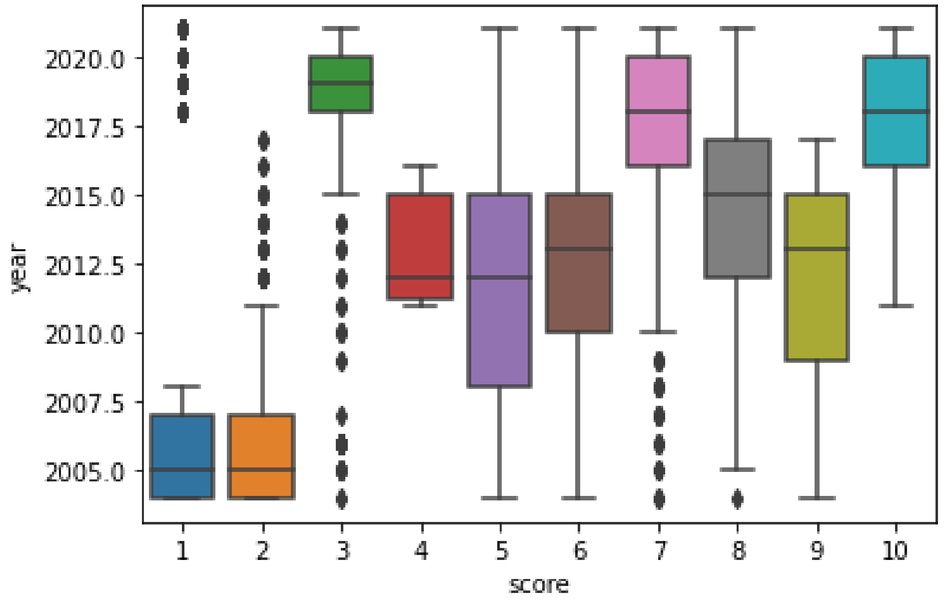
```
In [153]: ve_df.fuelType1.value_counts()

Out[153]: Regular Gasoline     15596
          Premium Gasoline     12175
          Diesel                 365
          Electricity            254
          Midgrade Gasoline      136
          Natural Gas             30
          Name: fuelType1, dtype: int64
```

We can combine midgrade and regular gasolines as 'Regular Gasoline'. We can also combine 'Electricity' and 'Natural Gas' as 'Green Fuel'.

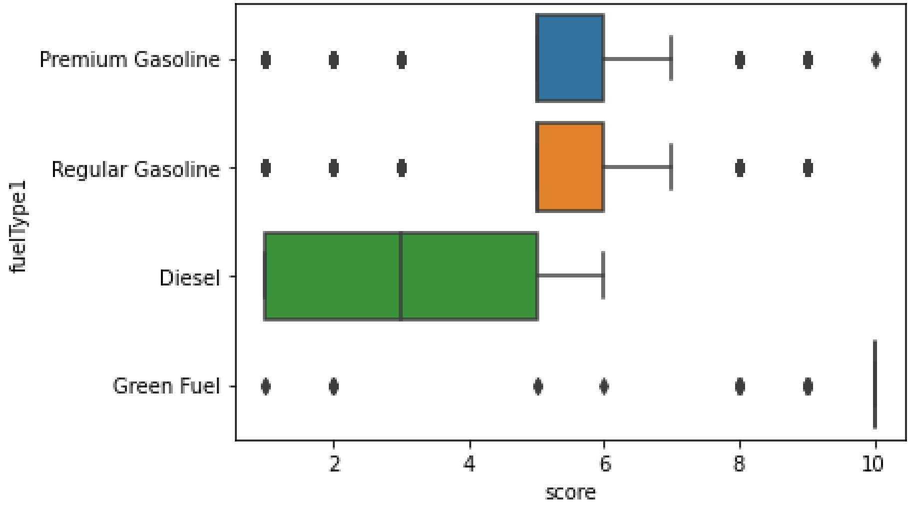
```
In [154]: for ft in ve_df.fuelType1.unique():
              selection = ve_df['fuelType1'] == ft
              if ft == 'Midgrade Gasoline':
                  ve_df.loc[selection, 'fuelType1'] = 'Regular Gasoline'
              elif ft in ['Electricity', 'Natural Gas']:
                  ve_df.loc[selection, 'fuelType1'] = 'Green Fuel'
          ve_df.fuelType1.value_counts()

Out[154]: Regular Gasoline    15732
          Premium Gasoline    12175
          Diesel                365
          Green Fuel            284
          Name: fuelType1, dtype: int64
```

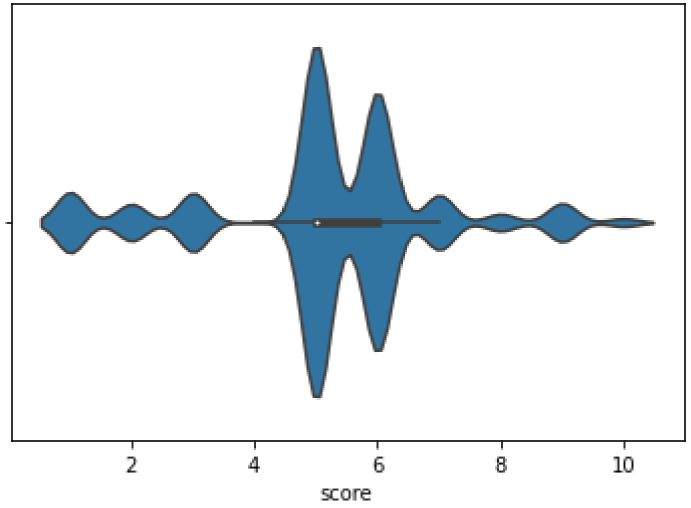
Correlation & Distribution Analysis

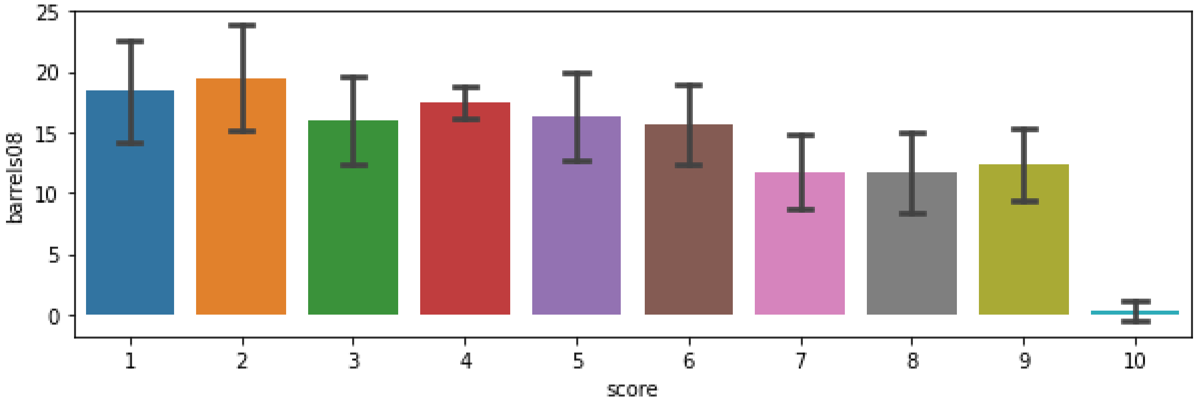




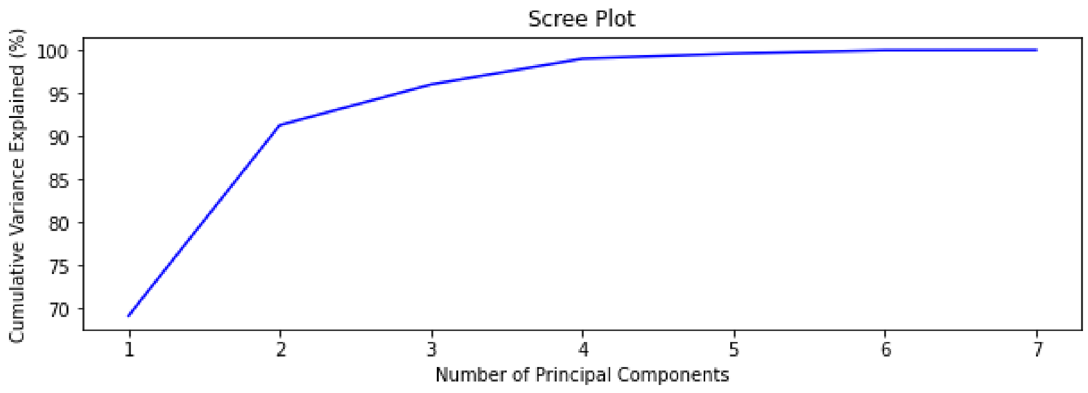
Correlation & Distribution Analysis

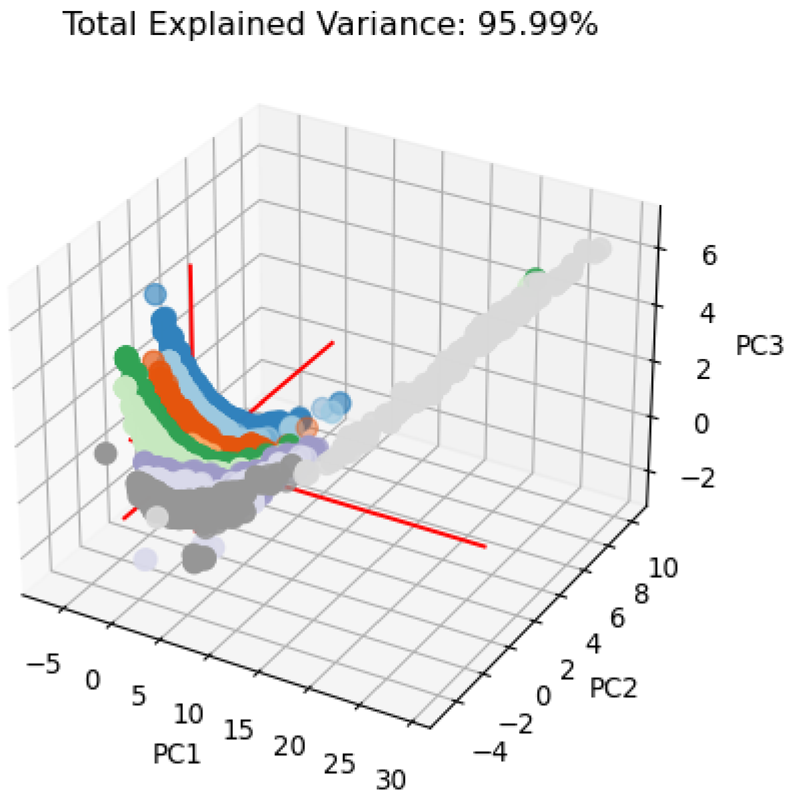






Principal Component Analysis





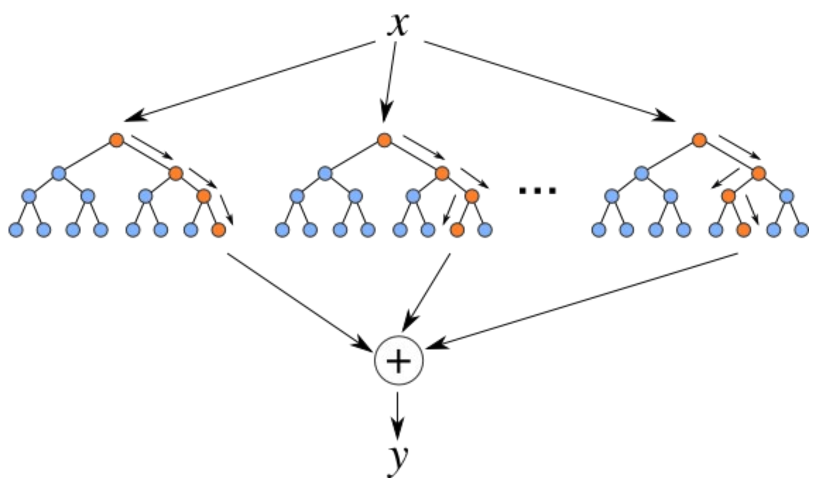
- 3-component solution explains 96% variance in the original data
- For predictive modeling, we thought of avoiding PCA

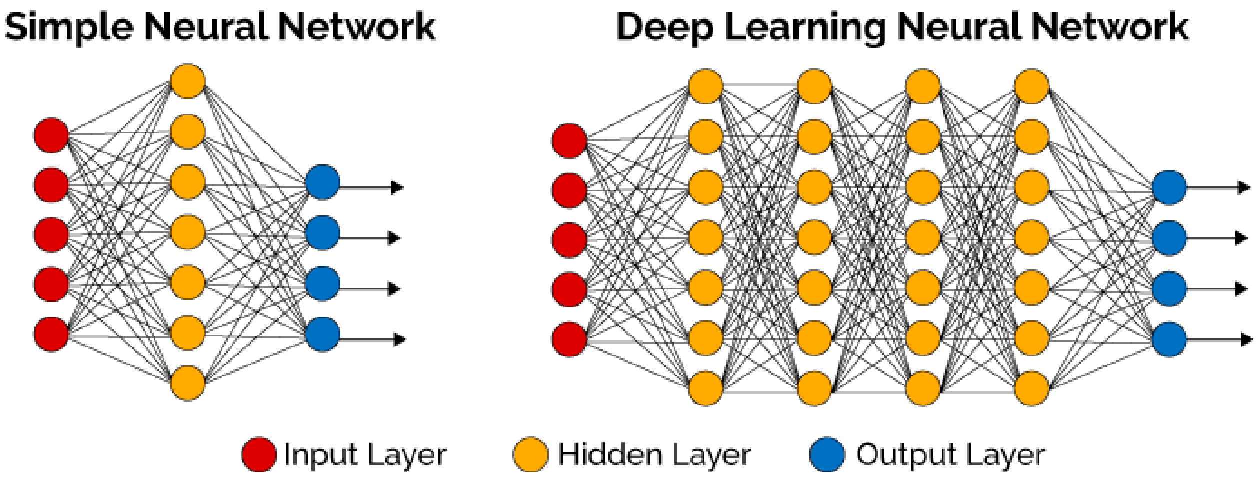
Predictive Modeling

1. Decision process: Classification

2. Error function: evaluate a model

3. Model optimization: adjusting model hyperparameters to get the best model





Source: *https://www.ibm.com/cloud/learn/machine-learning*

Machine Learning

- Ensemble Algorithms, Neural Network, SVM, Multinomial Logistic Regression
- Model Evaluation and Comparison
- Scalable ML using Dask

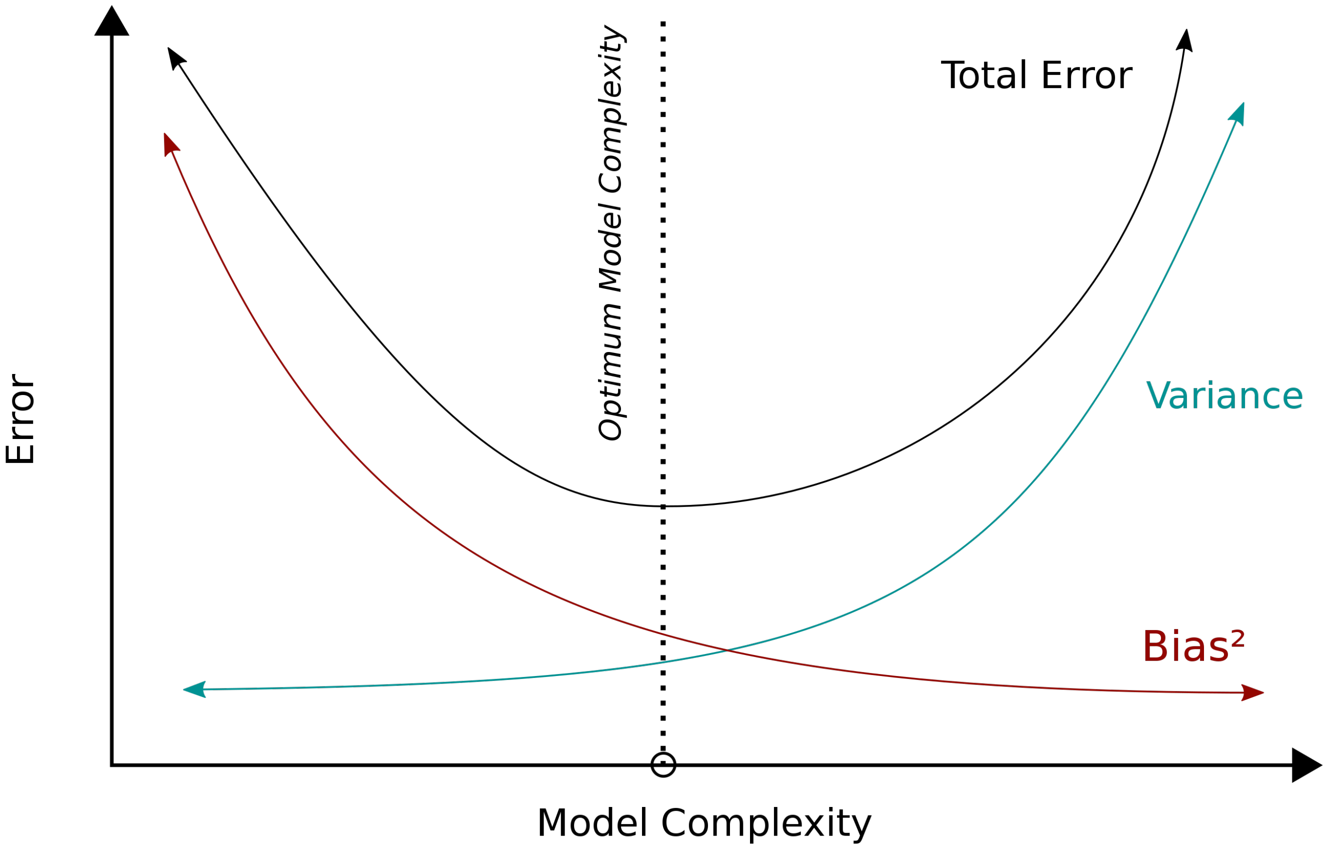


Image source: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Bias–Variance Tradeoff

Simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set

High bias: underfitting

High variance: overfitting

Regularization to control under or over fitting

Model Comparison

1. **Ensemble Learning:** Gradient Boosting Trees (GBT), Random Forests (RF), Boosted Random Forests (RF), Extremely Randomized Trees (ETR)

2. **Neural Network**: Multi-layer Perceptron Classifier (MLP)

3. **Support Vector Classifier (SVC)**

4. **Multinomial Logistic Regression (MLR)**

Note: 70% train, 30% test with stratified split, Train and Test predictors scaled using MinMaxScaler for all the models

Model Comparison: MLR

Out[219]:

	Data	F1 Score	Precision	Recall	Balanced Accuracy	Kappa Score
0	Train	0.331403	0.329090	0.509821	0.509821	0.189551
1	Validation	0.326589	0.326447	0.469016	0.483657	0.183054
2	Test	0.346034	0.353011	0.454485	0.409037	0.161838

In [220]:
```
print(report)
```

```
              precision    recall  f1-score   support

           1       0.13      0.31      0.18       584
           2       0.25      0.61      0.35       354
           3       0.32      0.53      0.40       566
           5       0.49      0.19      0.27      3418
           6       0.51      0.19      0.28      2506
           7       0.32      0.35      0.33       527
           8       0.06      0.49      0.11       164
           9       0.14      0.42      0.21       370
          10       0.96      1.00      0.98        76

   micro avg       0.27      0.27      0.27      8565
   macro avg       0.35      0.45      0.35      8565
weighted avg       0.42      0.27      0.28      8565
```

In [221]:
```
print(search_mlr.best_params_)
```

```
{'C': 1, 'max_iter': 300, 'solver': 'saga', 'tol': 0.0001}
```

Model Comparison: SVC

Out[20]:

	Data	F1 Score	Precision	Recall	Balanced Accuracy	Kappa Score
0	Train	0.370993	0.365296	0.577352	0.577352	0.231768
1	Validation	0.356233	0.355257	0.511651	0.532175	0.220769
2	Test	0.399593	0.386629	0.578398	0.578398	0.250525

In [21]:
```
print(report)
```

```
              precision    recall  f1-score   support

           1       0.46      0.46      0.46       584
           2       0.24      0.75      0.36       354
           3       0.41      0.71      0.52       566
           4       0.13      1.00      0.24         2
           5       0.63      0.23      0.34      3418
           6       0.50      0.30      0.38      2506
           7       0.33      0.54      0.41       527
           8       0.06      0.37      0.10       164
           9       0.14      0.43      0.21       370
          10       0.97      1.00      0.99        76

    accuracy                           0.36      8567
   macro avg       0.39      0.58      0.40      8567
weighted avg       0.50      0.36      0.37      8567
```

In [23]:
```
print(search_svm.best_params_)
```

```
{'C': 1, 'break_ties': True, 'gamma': 0.8, 'tol': 0.01}
```

Model Comparison: RF

Out[236]:

	Data	F1 Score	Precision	Recall	Balanced Accuracy	Kappa Score
0	Train	0.578722	0.555383	0.714078	0.714078	0.422408
1	Validation	0.436231	0.426816	0.533302	0.547823	0.263565
2	Test	0.408527	0.398950	0.420700	0.420700	0.227434

In [237]:
```
print(report)
```

```
              precision    recall  f1-score   support

           1       0.57      0.61      0.59       584
           2       0.40      0.45      0.43       354
           3       0.66      0.69      0.67       566
           4       0.00      0.00      0.00         2
           5       0.47      0.43      0.45      3418
           6       0.35      0.35      0.35      2506
           7       0.37      0.39      0.38       527
           8       0.12      0.19      0.14       164
           9       0.08      0.09      0.08       370
          10       0.97      1.00      0.99        76

    accuracy                           0.42      8567
   macro avg       0.40      0.42      0.41      8567
weighted avg       0.43      0.42      0.42      8567
```

In [238]:
```
print(search_rf.best_params_)
```

```
{'class_weight': 'balanced_subsample', 'criterion': 'entropy', 'max_depth': None, 'max_features': 10, 'max_samples': 0.5, 'min_
samples_leaf': 1, 'n_estimators': 500}
```

Model Comparison: GBT

Out[22]:

	Data	F1 Score	Precision	Recall	Balanced Accuracy	Kappa Score
0	Train	0.702728	0.644762	0.839862	0.839862	0.540943
1	Validation	0.481861	0.452456	0.556968	0.568588	0.324939
2	Test	0.514768	0.471515	0.633385	0.633385	0.329062

In [23]:
```
print(report)
```

```
              precision    recall  f1-score   support

           1       0.50      0.66      0.57       584
           2       0.36      0.67      0.47       354
           3       0.55      0.81      0.65       566
           4       0.50      1.00      0.67         2
           5       0.64      0.30      0.41      3418
           6       0.49      0.47      0.48      2506
           7       0.39      0.54      0.45       527
           8       0.15      0.48      0.23       164
           9       0.15      0.41      0.22       370
          10       0.97      1.00      0.99        76

    accuracy                           0.45      8567
   macro avg       0.47      0.63      0.51      8567
weighted avg       0.53      0.45      0.46      8567
```

In [24]:
```
print(search_dask_lgbm.best_params_)
```

```
{'boosting_type': 'gbdt', 'colsample_bynode': 1, 'colsample_bytree': 1, 'learning_rate': 0.05, 'max_bin': 127, 'max_depth': 10,
'min_child_samples': 20, 'min_data_in_bin': 3, 'n_estimators': 300, 'num_leaves': 31, 'path_smooth': 0.2, 'subsample': 1}
```

Model Comparison: BRF

Out[20]:

	Data	F1 Score	Precision	Recall	Balanced Accuracy	Kappa Score
0	Train	0.457200	0.431045	0.675145	0.675145	0.331778
1	Validation	0.424028	0.404417	0.578920	0.602269	0.296177
2	Test	0.449043	0.417397	0.644578	0.644578	0.309786

In [21]:
```
print(report)
```

```
              precision    recall  f1-score   support

           1       0.46      0.62      0.53       584
           2       0.33      0.72      0.45       354
           3       0.49      0.80      0.60       566
           4       0.10      1.00      0.17         2
           5       0.64      0.26      0.37      3418
           6       0.49      0.43      0.46      2506
           7       0.38      0.50      0.43       527
           8       0.14      0.66      0.23       164
           9       0.18      0.46      0.26       370
          10       0.97      1.00      0.99        76

    accuracy                           0.43      8567
   macro avg       0.42      0.64      0.45      8567
weighted avg       0.52      0.43      0.43      8567
```

In [22]:
```
print(search_brf.best_params_)
```

```
{'colsample_bynode': 1, 'colsample_bytree': 1, 'learning_rate': 0.05, 'max_bin': 255, 'max_depth': -1, 'min_child_samples': 20,
'min_data_in_bin': 3, 'n_estimators': 500, 'num_leaves': 63, 'path_smooth': 0, 'subsample': 0.9}
```

Model Comparison: ETR

Out[33]:

	Data	F1 Score	Precision	Recall	Balanced Accuracy	Kappa Score
0	Train	0.533500	0.514130	0.709851	0.709851	0.401488
1	Validation	0.400826	0.395748	0.519259	0.537028	0.239507
2	Test	0.462869	0.428880	0.533174	0.533174	0.221032

In [34]:

```python
print(report)
```

```
              precision    recall  f1-score   support

           1       0.55      0.61      0.58       584
           2       0.39      0.53      0.45       354
           3       0.62      0.67      0.64       566
           4       0.40      1.00      0.57         2
           5       0.46      0.36      0.40      3418
           6       0.36      0.36      0.36      2506
           7       0.34      0.40      0.37       527
           8       0.12      0.24      0.16       164
           9       0.10      0.16      0.12       370
          10       0.96      1.00      0.98        76

    accuracy                           0.40      8567
   macro avg       0.43      0.53      0.46      8567
weighted avg       0.42      0.40      0.40      8567
```

In [35]:

```python
print(search_et.best_params_)
```

```
{'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': None, 'max_features': 10, 'max_samples': 0.8, 'min_samples_le
af': 1, 'n_estimators': 300}
```

Model Comparison: MLP

Out[21]:

	Data	F1 Score	Precision	Recall	Balanced Accuracy	Kappa Score
0	Train	0.526306	0.590153	0.513441	0.513441	0.385971
1	Validation	0.450087	0.489226	0.445315	0.453457	0.324013
2	Test	0.483473	0.530263	0.476535	0.428881	0.326520

In [22]:

```
print(report)
```

```
              precision    recall  f1-score   support

           1       0.57      0.59      0.58       584
           2       0.49      0.38      0.43       354
           3       0.57      0.72      0.64       566
           5       0.52      0.67      0.59      3418
           6       0.48      0.38      0.43      2506
           7       0.53      0.37      0.43       527
           8       0.39      0.09      0.15       164
           9       0.25      0.09      0.13       370
          10       0.97      1.00      0.99        76

   micro avg       0.52      0.52      0.52      8565
   macro avg       0.53      0.48      0.48      8565
weighted avg       0.51      0.52      0.50      8565
```

In [23]:

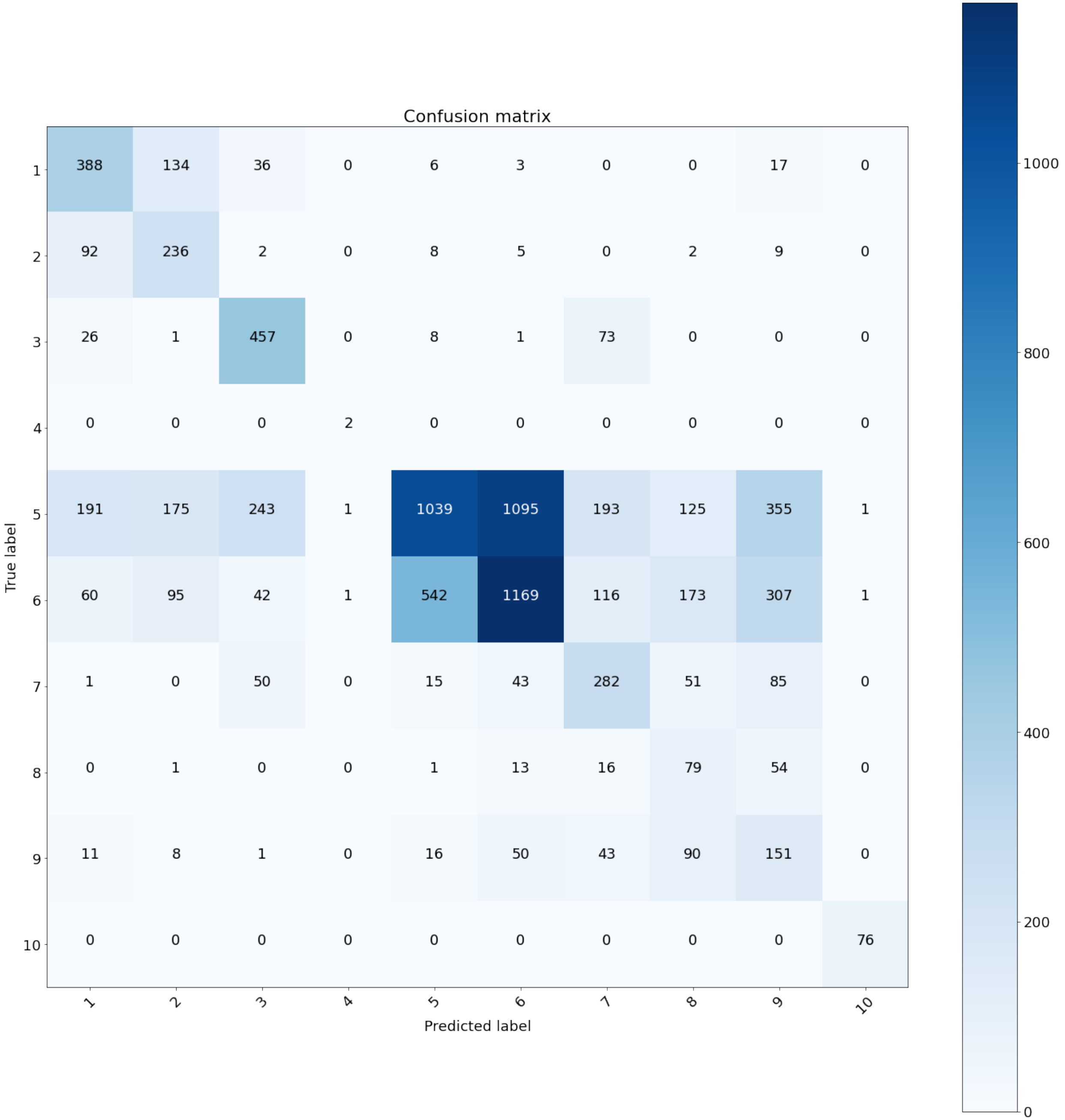
```
print(search_mlp.best_params_)
```

```
{'hidden_layer_sizes': (512, 256), 'learning_rate_init': 0.001, 'max_iter': 100}
```

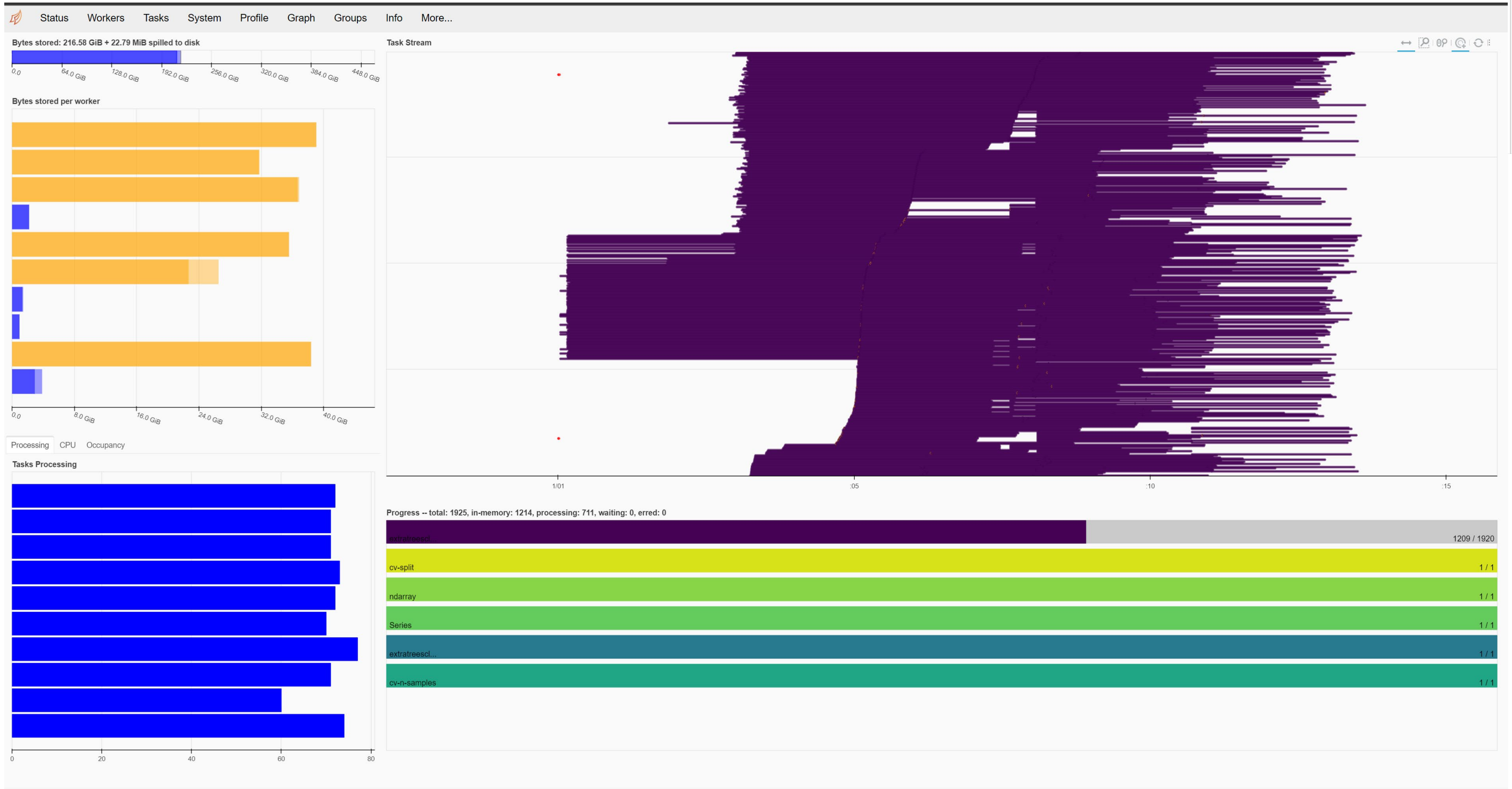
Final Model Results: GBT

ROC_AUC	0.622634
F1 Score	0.514768
Precision	0.471515
Recall	0.633385
Balanced Accuracy	0.633385
Kappa Score	0.329062

Test Data



Dask + Foundry



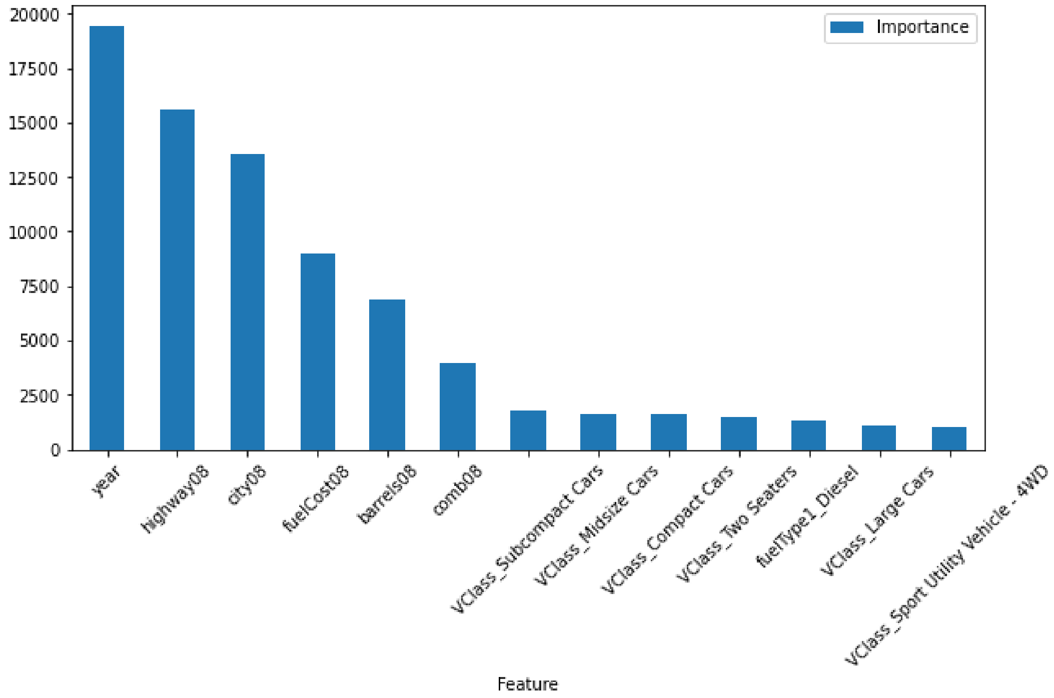

```python
param_dict = {
    'n_estimators': [300, 500],
    'max_features': [3, 5, 7, 10],
    'max_depth': [6, 10, None],
    'max_samples': [0.8, 0.5, 0.9, None],
    'min_samples_leaf': [1],
    'criterion': ['gini', 'entropy'],
    'class_weight': ['balanced', 'balanced_subsample']
}
cv = RepeatedKFold(n_splits=5, n_repeats=1, random_state=random_state)
model = ExtraTreesClassifier(
    class_weight='balanced',
    random_state=random_state,
    n_jobs=-1
)
model_grid = DaskGCV(
    estimator=model,
    param_grid=param_dict,
    scoring=scoring_metrics,
    n_jobs=-1, cv=cv, refit=scoring_metrics['f1_' + score_avg],
    return_train_score=True
)
```

ETR Training:

- 216 GB memory!
- 40 cores
- 10 nodes
- ~10 mins of GridSearch

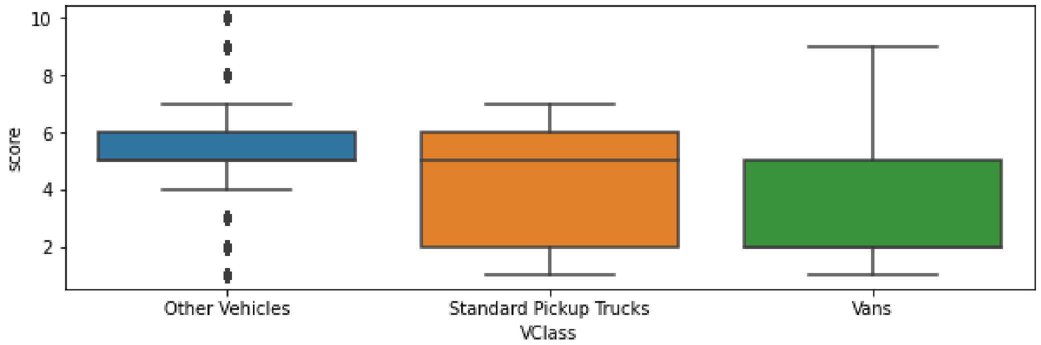
Revisiting Research Questions

- Which predictors are most suited for predicting the EPA emission score?



Revisiting Research Questions

- Which machine learning model works best for this data set? Gradient Boosting Trees
- Which vehicle class has poor emission ratings? Pickup Trucks and Vans



Conclusions

- We were able to successfully answer the research questions
- 7 classification models are compared
- Correlation analysis and PCA-based dimensionality reduction were not helpful (possibly due to strong non-linearity)
- Additional data are required to improve the model results and address class imbalance.
- With Dask and Missouri S&T Foundry High-Performance Computing, we could save hours/days of computational time : Scalable ML is the way to go!
- This kind of modeling and analysis, if pushed to a production environment, could enable policymakers to address critical issues in the sustainable transportation industry.

Thank you!