IDS 575 Machine Learning Statistics

AIR QUALITY INDEX PREDICTION

Amin Abbasi | Rishitha Addagada | Vipul Singh | Kate Veach

AIR QUALITY IMPORTANCE

AQI

Standardized ranking system to communicate air quality in a specific location

HEALTH RISKS

Cardiovascular, respiratory, allergies, & increased mortality rates

ENVIRONMENTAL

Acid rain, smog, greenhouse gases & climate change

ECONOMIC

Increased healthcare costs, decreased workplace productivity & tourism

HABITATS

Habitat contamination such as oceans & forests

HUMANS

Children, elderly, & sensitive groups at greatest risk

PROBLEM STATEMENT

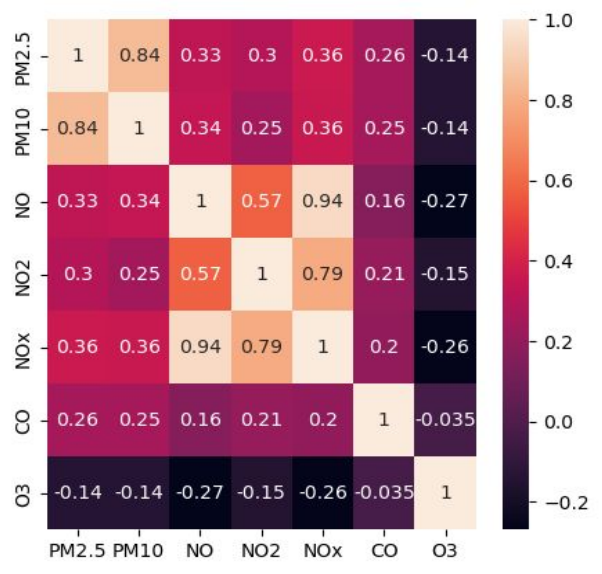
- **AQI is a time series problem where the final AQI value is calculated based on the concentration of all pollutants, and results are forecasted based on the final AQI value**
    - Determine final label and convert into a classification model
- <u>Problem</u>: the necessity to determine the saturation point of each pollutant
    - Input the given concentration along with the saturation point value to derive AQI for a pollutant
        - Area AQI values are derived using an empirical formula, taking into account individual AQI values
- <u>Solution</u>: Generate a model where we can easily find the AQI index of an area when given the concentration of pollutant
    - Concentration can easily be extracted using particulate monitor
- We used different machine learning methods and compared several models using confusion matrix, classification report, and ROC curve to evaluate Precision, Recall, F-Score and other relevant matrices.

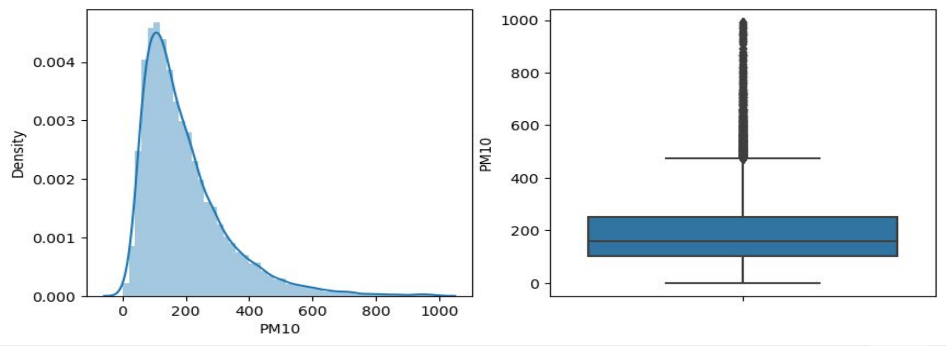
DATA DESCRIPTION

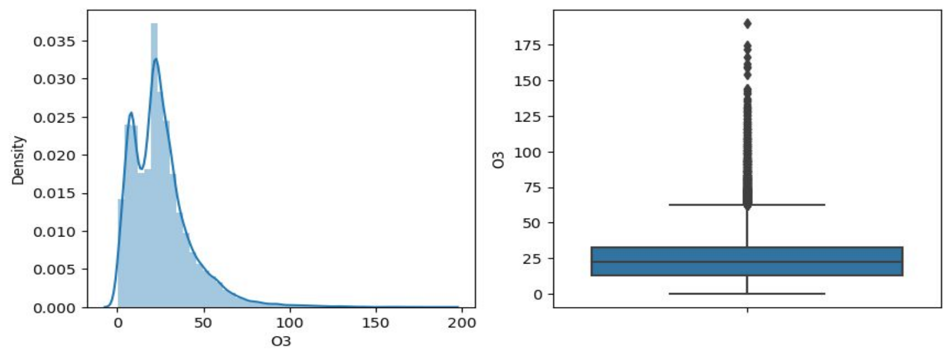
- Historical data collected hourly from 2015-2020 of the 7 most common pollutants: **PM2.5, PM10, NO, NO2, NOx, O3, CO**
- Dataset source: Kaggle
- Shape: 18,205 x 11
- Classes: ‘Good’, ‘Moderate’, ‘Poor’
- Class distribution:
  - Good: 15.6%
  - Moderate: 43.76%
  - Poor: 40.63%

	StationId	Datetime	PM2.5	PM10	NO	NO2	NOx	CO	O3	AQI	AQI_Bucket
0	DL012	26/09/17 16:00	114.33	138.57	15.40	30.73	0.0	0.90	73.94	164.0	Moderate
1	DL012	26/09/17 17:00	137.00	162.35	15.17	31.08	0.0	0.88	77.51	175.0	Moderate
2	DL012	26/09/17 18:00	109.99	135.96	15.42	30.70	0.0	0.90	73.76	180.0	Moderate
3	DL012	26/09/17 19:00	72.24	84.96	16.89	28.75	0.0	1.01	58.50	178.0	Moderate
4	DL012	26/09/17 20:00	48.96	58.08	22.77	23.74	0.0	1.45	33.71	173.0	Moderate

EXPLORATORY DATA ANALYSIS





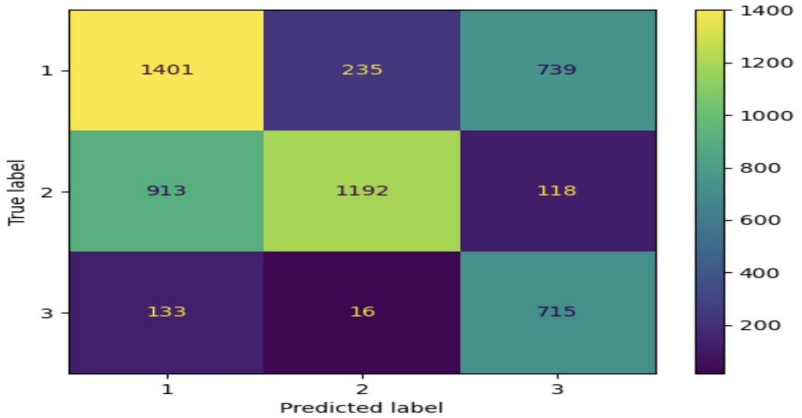


BASELINE MODEL: GAUSSIAN NAIVE BAYES

- Selected due to each pollutant having an independent capacity to predict the output variable
- Distribution of some pollutants is not bell shaped, leading to imperfect predictions
- Code: model1=GaussianNB()
- Output:

Classification Report

	precision	recall	f1-score	support
1	0.57	0.59	0.58	2375
2	0.83	0.54	0.65	2223
3	0.45	0.83	0.59	864
accuracy			0.61	5462
macro avg	0.62	0.65	0.61	5462
weighted avg	0.66	0.61	0.61	5462



MODEL 1: K-NEAREST NEIGHBOR (KNN)

- KNN is a non-parametric method that utilizes proximity to perform calculation
- Model fine tuning: **GridSearchCV**
  - Included all possible parameters of KNN
  - Utilized cross validation techniques to determine best parameters
- Outcome:
  - Increase in precision rates for all classes
  - Reduction in both false positive and false negative rates

```python
params = {
    'n_neighbors': range(1, 31),
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski']
}
```

```python
from sklearn.model_selection import GridSearchCV
```

```python
grid_search = GridSearchCV(KNeighborsClassifier(), params, cv=5, verbose=1, scoring='accuracy')
```

```python
grid_search.fit(X_train, y_train)
```

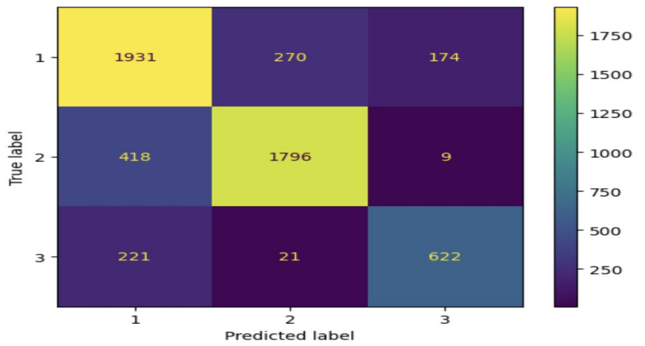
```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
```

```
GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier
KNeighborsClassifier()
```

```python
print("Best parameters:", grid_search.best_params_)
```

```
Best parameters: {'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance'}
```

```
Classification Report
              precision    recall  f1-score   support

           1       0.75      0.81      0.78      2375
           2       0.86      0.81      0.83      2223
           3       0.77      0.72      0.75       864

    accuracy                           0.80      5462
   macro avg       0.79      0.78      0.79      5462
weighted avg       0.80      0.80      0.80      5462
________________________________________________________
```



MODEL 2: LOGISTIC REGRESSION (LR)

- LR is very effective for classifying multi-class labels
- Model fine tuning:
  - Utilized softmax function instead of the sigmoid function for multi-class classification
  - Predicted outcome (y) determined using the argmax of the probabilities obtained for all classes

```python
from sklearn.linear_model import LogisticRegression
```

```python
para_lr={'multi_class':['multinomial'], 'C':np.logspace(-4, 4, 20),
          'solver':['lbfgs', 'newton-cg', 'sag', 'saga'],
         'class_weight':['balanced'],'penalty': ['l2','l1','elasticnet']}
```

```python
grid_search_lr = GridSearchCV(LogisticRegression(), para_lr,cv=15, verbose=1, scoring='accuracy')
```

```python
grid_search_lr.fit(X_train, y_train)
```

```
Fitting 15 folds for each of 240 candidates, totalling 3600 fits
```

```
GridSearchCV
estimator: LogisticRegression
LogisticRegression
```

```python
print("Best parameters:", grid_search_lr.best_params_)
```

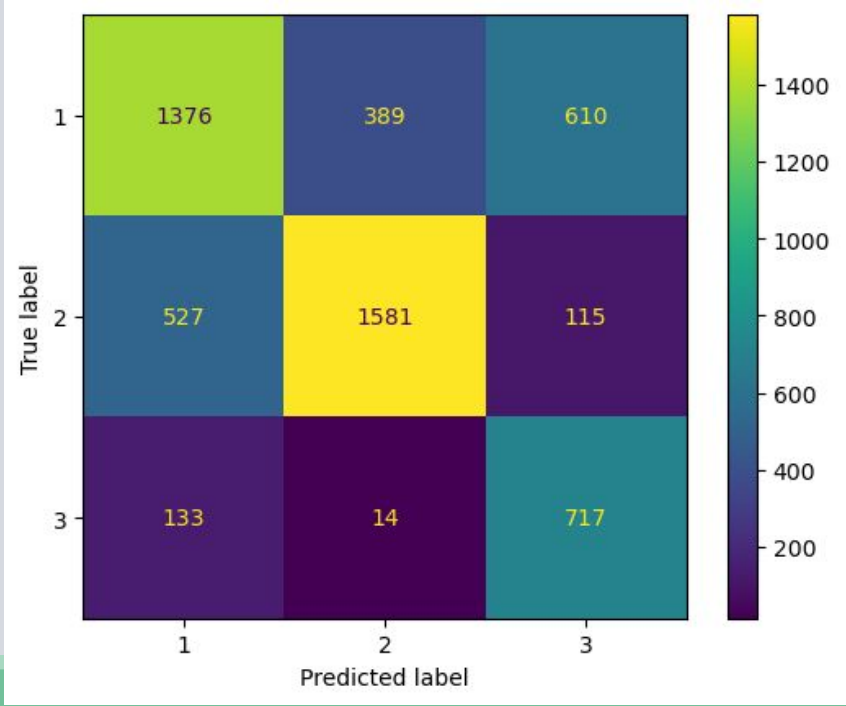
```
Best parameters: {'C': 11.288378916846883, 'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l
2', 'solver': 'saga'}
```

MODEL 2: LOGISTIC REGRESSION (LR)

- The output is not satisfactory:
    - The accuracy, precision, and F-1 scores decrease compared to KNN
- This could be due to our dataset not being suitable for LR or the assumptions of LR are not satisfied

Classification Report

	precision	recall	f1-score	support
1	0.68	0.58	0.62	2375
2	0.80	0.71	0.75	2223
3	0.50	0.83	0.62	864
accuracy			0.67	5462
macro avg	0.66	0.71	0.67	5462
weighted avg	0.70	0.67	0.68	5462



MODEL 2: LR SMOTE PROCESS

- We implemented the SMOTE process because one of our classes is a minority
  - SMOTE generates synthetic samples for the minority class to handle imbalanced data
- With SMOTE, we observed some improvement in the prediction of the LR model:

```
from imblearn.over_sampling import SMOTE
```

```
y_sm=df_t_sm[['AQI_Bucket']]
```

```
sm=SMOTE()
```

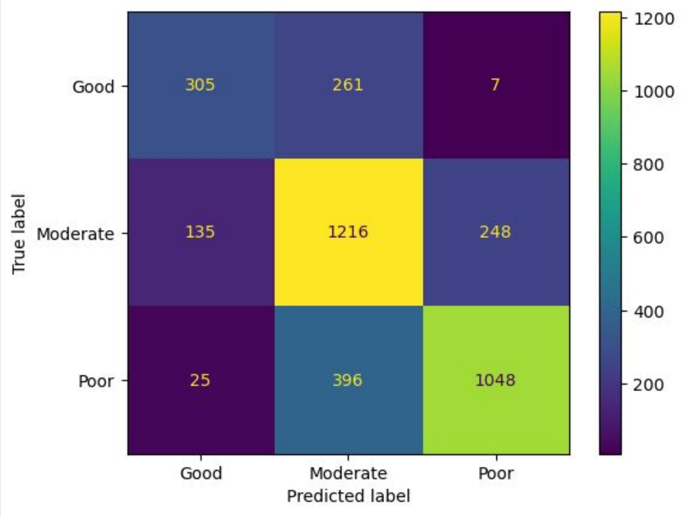
```
X_sm,y_sm=sm.fit_resample(X_sm,y_sm)
```

```
y_sm.value_counts()
```

```
AQI_Bucket
1          7968
2          7968
3          7968
Name: count, dtype: int64
```

Classification Report

	precision	recall	f1-score	support
1	0.59	0.57	0.58	2427
2	0.79	0.70	0.75	2332
3	0.73	0.84	0.78	2413
accuracy			0.70	7172
macro avg	0.70	0.70	0.70	7172
weighted avg	0.70	0.70	0.70	7172

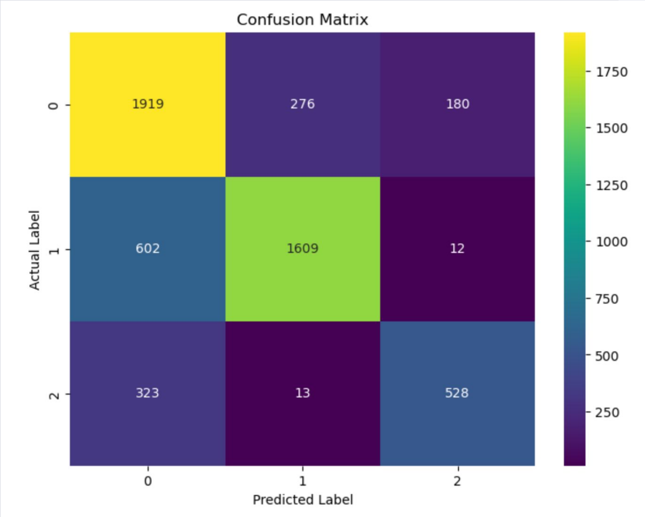


EXPERIMENTAL RESULTS & PREDICTIONS

- **Simple SVM Model**:
  - Utilized to provide a baseline point for comparison using a “one-vs-all” strategy
  - Trained a basic SVM model with a RBF kernel
  - <u>Output</u>: 74% accuracy

Classification Report:

	precision	recall	f1-score	support
0	0.67	0.81	0.74	2375
1	0.85	0.72	0.78	2223
2	0.73	0.61	0.67	864
accuracy			0.74	5462
macro avg	0.75	0.71	0.73	5462
weighted avg	0.75	0.74	0.74	5462

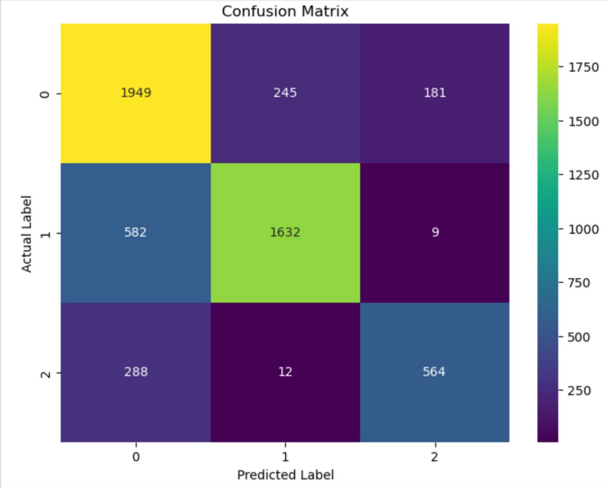


EXPERIMENTAL RESULTS & PREDICTIONS

- **Grid Search for Hyperparameter Tuning**:
    - Demonstrates improvements in accuracy through hyperparameter tuning
    - Used grid search cross-validation to find the best hyperparameters for the SVM model with the “one-vs-all” strategy
    - <u>Best Parameters</u>: {‘C’:10, ‘gamma’:’scale’, ‘kernel’:’rbf’}
    - <u>Output</u>: 76% accuracy

Classification Report for Grid Search with Hyperparameter Tuning:

	precision	recall	f1-score	support
0	0.69	0.82	0.75	2375
1	0.86	0.73	0.79	2223
2	0.75	0.65	0.70	864
accuracy			0.76	5462
macro avg	0.77	0.74	0.75	5462
weighted avg	0.77	0.76	0.76	5462



MODEL COMPARISON BASED ON ACCURACY

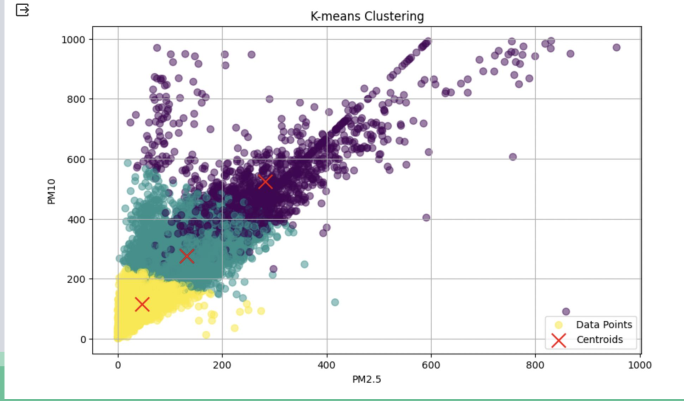
MODEL	ACCURACY
Gaussian Naive Bayes (Baseline)	61%
K-Nearest Neighbor (KNN)	80%
Logistic Regression	67%
Logistic Regression with SMOTE	70%
Logistic Regression with Stacking	79%
SVM	74%
SVM Grid Search Hyperparameter Tuning	76%

UNSUPERVISED LEARNING MODEL:
K-MEANS CLUSTERING

- Implemented K-Means Clustering to gain insights on underlying structure of our dataset, particularly PM2.5 and PM10
- Goal: identify inherent clusters or groupings that could potentially reveal patterns related to air quality characteristics
  - Allowing us to partition the data into distinct clusters based on similarity in air quality parameters

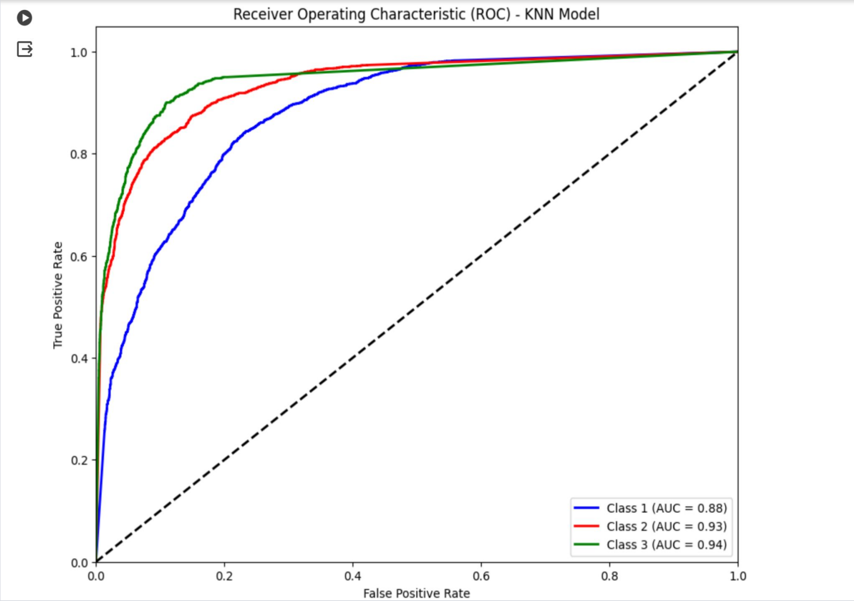
```python
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(df_exp_AQI_Bucket)
plt.figure(figsize=(10, 6))

plt.scatter(df_exp_AQI_Bucket['PM2.5'], df_exp_AQI_Bucket['PM10'], c=clusters, cmap='viridis', s=50, alpha=0.5, label='Data Points')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='x', s=200, label='Centroids')
```

```
Silhouette Score: 0.4371433264797495
```



OBSERVATIONS FROM BEST PERFORMING MODEL

- ROC Curve of our KNN Model
  - The 3 curves represent the performance of KNN for each class
  - AUC is listed for each class
    - Class 1 AUC=0.88
    - Class 2 AUC=0.93
    - Class 3 AUC=0.94



ANY QUESTIONS?