Disparities in Air Pollution Exposure

ZHUOPEI JIN | DANIEL SUNG | ALYCE KEYS

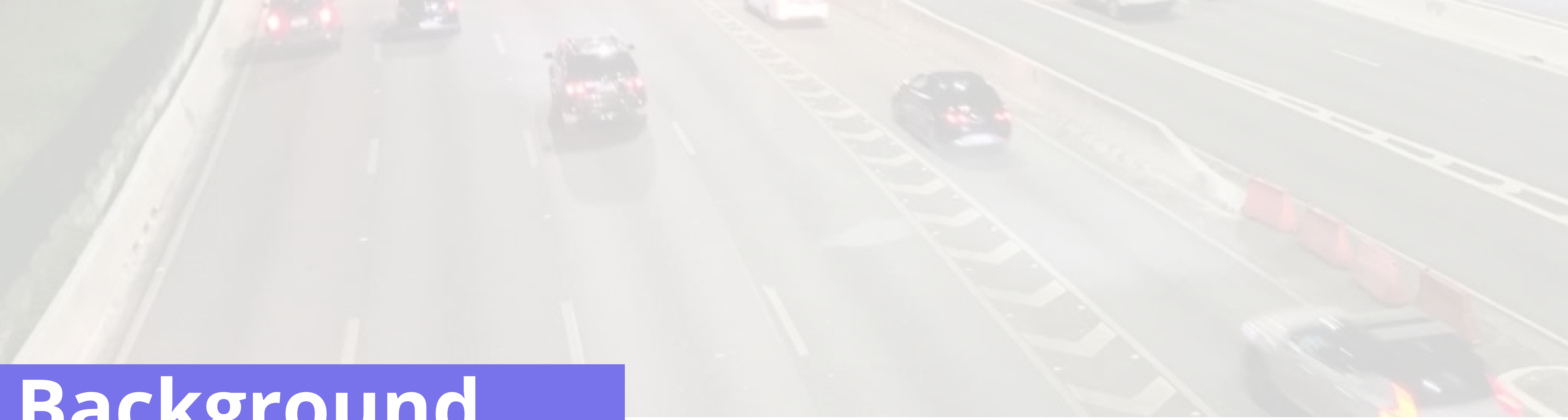
01 Background

02 Problem Statement

03 Method and Data

04 Results and Discussion

05 Conclusion



Background

Air quality is the measure of how polluted the air is in a particular area

Air pollution is a major environmental risk to public health and can result in health issues including heart disease, lung and bronchus cancer, stroke, and death

PM2.5 are fine particles less than 2.5 micrometers in diameter that pose risk to health and visibility

Problem Statement

It is well documented that racial/ethnic minorities and people of low socioeconomic status in the US are at a higher risk of death from being exposed to PM2.5 (Di, Q. et al. 2017)

We aim to explore how different demographics are affected by PM2.5 in their respective regions in Texas

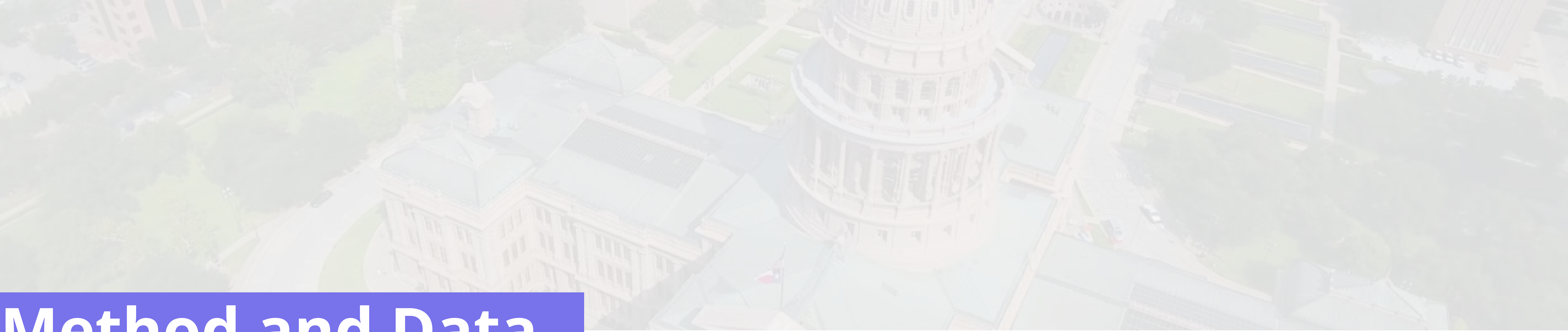
We then plan to use these models to predict

Method and Data

ZCTA

Demographic Data

Environmental Data



Method and Data

ZCTA

ZCTA: ZIP Code Tabulation Area

Effectively captured popluation patterns using tracts

	Unnamed: 0	year	zcta	pm25
789255	789256	2014	99921.0	NaN
789256	789257	2015	99921.0	NaN
789257	789258	2016	99921.0	NaN
789258	789259	2017	NaN	NaN
789259	789260	2018	NaN	NaN

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 789260 entries, 0 to 789259
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   year    789260 non-null  int64  
 1   zcta    686681 non-null  Int64  
 2   pm25    689745 non-null  float64
dtypes: Int64(1), float64(1), int64(1)
memory usage: 18.8 MB
```

Method and Data

Demographic Data

Social Explorer, ACS(5 years) data from 2011 to 2021

- Total population
- Race
- Median Household Income
- Education Attainment

```python
#ACS Yearly Data
```

```python
acs_2011 = pd.read_csv('ACS 2011.csv')
acs_2011['year'] = 2011
acs_2012 = pd.read_csv('ACS 2012.csv')
acs_2012['year'] = 2012
acs_2013 = pd.read_csv('ACS 2013.csv')
acs_2013['year'] = 2013
acs_2014 = pd.read_csv('ACS 2014.csv')
acs_2014['year'] = 2014
acs_2015 = pd.read_csv('ACS 2015.csv')
acs_2015['year'] = 2015
acs_2016 = pd.read_csv('ACS 2016.csv')
acs_2016['year'] = 2016
acs_2017 = pd.read_csv('ACS 2017.csv')
acs_2017['year'] = 2017
acs_2018 = pd.read_csv('ACS 2018.csv')
acs_2018['year'] = 2018
acs_2019 = pd.read_csv('ACS 2019.csv')
acs_2019['year'] = 2019
acs_2020 = pd.read_csv('ACS 2020.csv')
acs_2020['year'] = 2020
acs_2021 = pd.read_csv('ACS 2021.csv')
acs_2021['year'] = 2021
```

```python
merged_acs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 346566 entries, 0 to 364973
Data columns (total 12 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   Geo_FIPS                     346566 non-null  int64  
 1   Geo_GEOID                    346566 non-null  object 
 2   zcta                         346566 non-null  int32  
 3   Median Household Income      346566 non-null  float64
 4   total population             346566 non-null  int64  
 5   population Density           346566 non-null  float64
 6   white_pop                    346566 non-null  int64  
 7   black_pop                    346566 non-null  int64  
 8   asian_pop                    346566 non-null  int64  
 9   hispanic_pop                 346566 non-null  int64  
 10  year                         346566 non-null  int64  
 11  Bachelor’s degree or higher  346566 non-null  int64  
dtypes: float64(2), int32(1), int64(8), object(1)
memory usage: 33.1+ MB
```

Method and Data

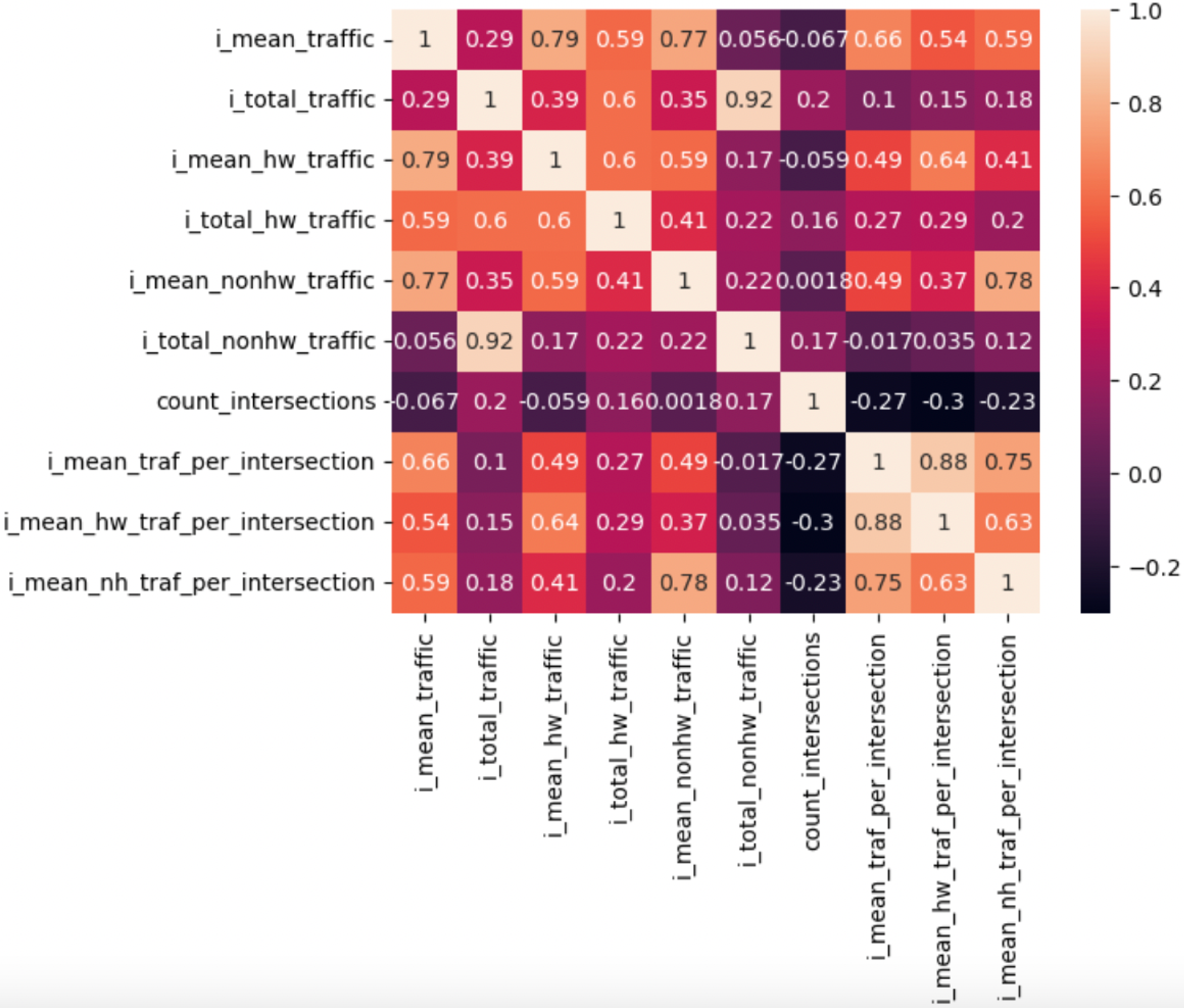
Environmental Data

- PM2.5 Concentrations (Target Variable)
- Traffic Volume

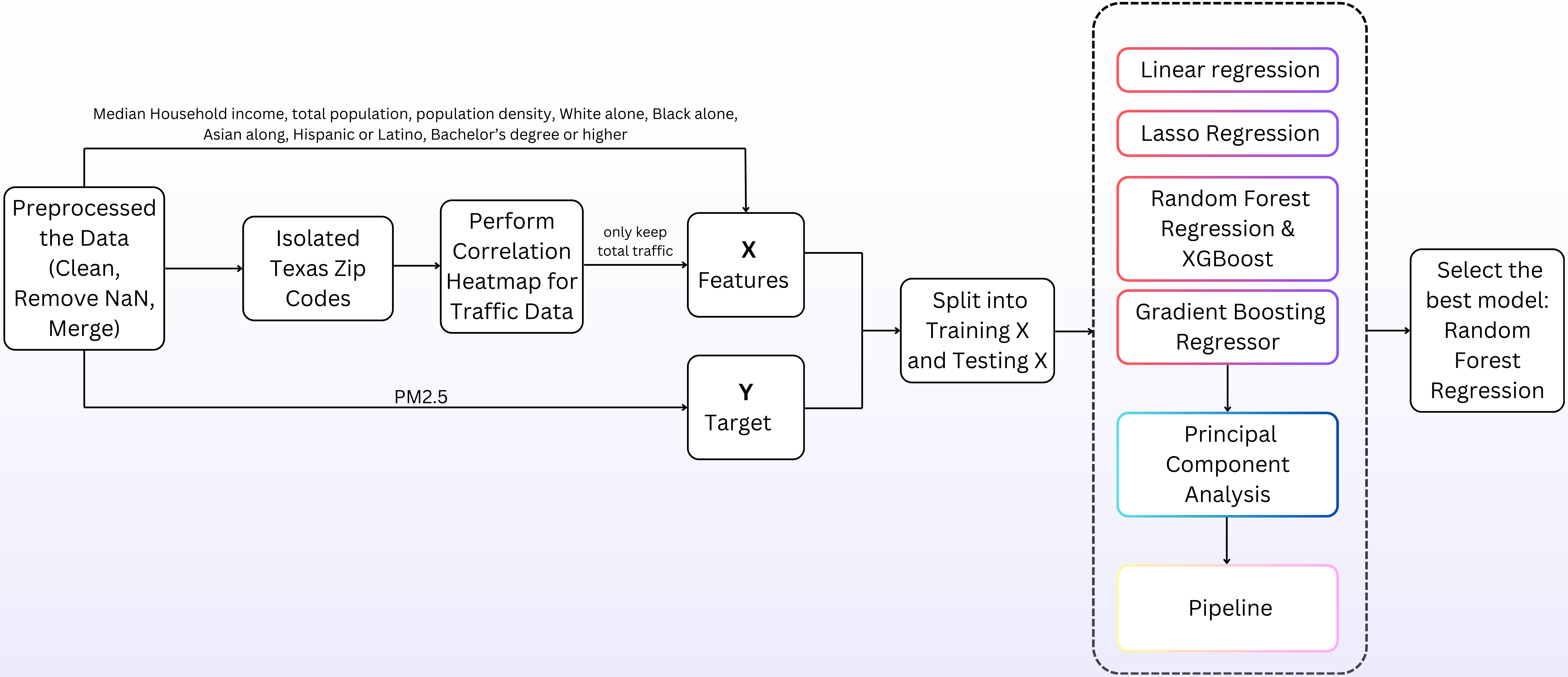
```
traffic
```

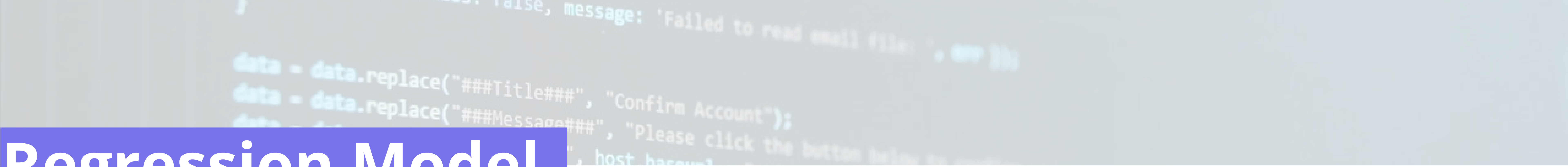
	zcta19	year	intp_flag	i_mean_traffic	i_min_traffic	i_max_traffic	i_count_traffic	i_total_traffic	i_mean_hw_traffic	i_min_hw_traffic	...	i_total_hw_traffi
0	1001	1993	0	25872.0000	25872.0000	25872.000	1.0	25872.00	NaN	NaN	...	0.
1	1001	1994	0	26000.0000	26000.0000	26000.000	1.0	26000.00	NaN	NaN	...	0.
2	1001	1995	0	15700.0000	4500.0000	23000.000	5.0	78500.00	NaN	NaN	...	0.
3	1001	1996	0	40600.0000	40600.0000	40600.000	1.0	40600.00	NaN	NaN	...	0.
4	1001	1997	0	32466.6700	9400.0000	56600.000	3.0	97400.00	NaN	NaN	...	0.
...	...	...	...	...	...	...	...	...	...	...	...	.
675321	99929	2014	1	891.6190	331.3333	1852.667	9.0	7648.00	NaN	NaN	...	0.
675322	99929	2015	1	899.4643	371.5000	1801.750	7.5	6428.25	NaN	NaN	...	0.
675323	99929	2016	1	907.3095	411.6667	1750.833	6.0	5208.50	NaN	NaN	...	0.
675324	99929	2017	1	915.1548	451.8333	1699.917	4.5	3988.75	NaN	NaN	...	0.
675325	99929	2018	0	923.0000	492.0000	1649.000	3.0	2769.00	NaN	NaN	...	0.

675326 rows × 22 columns


i_mean_traffic 1 0.29 0.79 0.59 0.77 0.056 -0.067 0.66 0.54 0.59
i_total_traffic 0.29 1 0.39 0.6 0.35 0.92 0.2 0.1 0.15 0.18
i_mean_hw_traffic 0.79 0.39 1 0.6 0.59 0.17 -0.059 0.49 0.64 0.41
i_total_hw_traffic 0.59 0.6 0.6 1 0.41 0.22 0.16 0.27 0.29 0.2
i_mean_nonhw_traffic 0.77 0.35 0.59 0.41 1 0.22 0.0018 0.49 0.37 0.78
i_total_nonhw_traffic 0.056 0.92 0.17 0.22 0.22 1 0.17 -0.017 0.035 0.12
count_intersections -0.067 0.2 -0.059 0.16 0.0018 0.17 1 -0.27 -0.3 -0.23
i_mean_traf_per_intersection 0.66 0.1 0.49 0.27 0.49 -0.017 -0.27 1 0.88 0.75
i_mean_hw_traf_per_intersection 0.54 0.15 0.64 0.29 0.37 0.035 -0.3 0.88 1 0.63
i_mean_nh_traf_per_intersection 0.59 0.18 0.41 0.2 0.78 0.12 -0.23 0.75 0.63 1
i_mean_traffic
i_total_traffic
i_mean_hw_traffic
i_total_hw_traffic
i_mean_nonhw_traffic
i_total_nonhw_traffic
count_intersections
i_mean_traf_per_intersection
i_mean_hw_traf_per_intersection
i_mean_nh_traf_per_intersection
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Research Methodology





Regression Model

Linear regression

Lasso Regression

Random Forest Regression & XGBoost

Gradient Boosting Regressor

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   pm25   R-squared:                       0.413
Model:                            OLS   Adj. R-squared:                  0.412
Method:                 Least Squares   F-statistic:                     352.5
Date:                Tue, 05 Dec 2023   Prob (F-statistic):               0.00
Time:                        12:34:04   Log-Likelihood:                -7675.5
No. Observations:                4516   AIC:                         1.537e+04
Df Residuals:                    4506   BIC:                         1.544e+04
Df Model:                           9
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
const                       6.7805      0.072     94.347      0.000       6.640       6.921
i_total_traffic           4.96e-07   5.76e-08      8.612      0.000    3.83e-07    6.09e-07
Median Household Income  2.234e-06     1.4e-06      1.601      0.110   -5.02e-07    4.97e-06
total population           -0.0005   9.37e-05     -5.378      0.000      -0.001      -0.000
population Density          0.0004   1.58e-05     28.397      0.000       0.000       0.000
white_pop                   0.0005   9.66e-05      5.659      0.000       0.000       0.001
black_pop                   0.0006   9.54e-05      5.838      0.000       0.000       0.001
asian_pop                   0.0005      0.000      5.123      0.000       0.000       0.001
hispanic_pop                0.0005    9.4e-05      5.631      0.000       0.000       0.001
Bachelor’s degree or higher -3.437e-05   8.54e-06     -4.027      0.000   -5.11e-05   -1.76e-05
==============================================================================
Omnibus:                      128.353   Durbin-Watson:                   2.016
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              247.123
Skew:                           0.203   Prob(JB):                     2.18e-54
Kurtosis:                       4.071   Cond. No.                     1.86e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.86e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model Overview:

R-squared of 41.0% indicates the model explains variability in pm25.

The overall model is statistically significant.

Predictor Variable Analysis

Every Daily Traffic Counts Positive

A one-unit increase is associated with a 5.151e-07 increase in pm25

Median Household Income Positive

Coefficient is positive but not statistically significant (p = 0.076).

Total Population Negative

A one-unit increase is associated with a 5.151e-07 increase in pm25

Population Density Positive

A one-unit increase is associated with a 0.0004 increase in pm25

White, Black, Asian, Hispanic Population Positive

Each one-unit increase is associated with an increase in pm25 by the respective coefficients.

Bachelor’s Degree or Higher Negative

A one-unit increase is associated with a decrease in pm25 by -2.919e-05.

Regression Model

- Linear regression
- **Lasso Regression**
- Random Forest Regression & XGBoost
- Gradient Boosting Regressor

	Feature	Coefficient
0	const	0.000000e+00
1	i_total_traffic	5.266859e-07
2	Median Household Income	2.324231e-06
3	total population	-2.438346e-05
4	population Density	4.387320e-04
5	white_pop	5.281370e-05
6	black_pop	6.950953e-05
7	asian_pop	1.798520e-05
8	hispanic_pop	4.831725e-05
9	Bachelor’s degree or higher	-3.155886e-05

Predictor Variable Analysis

Every Daily Traffic Counts	Positive
Median Household Income	Positive
Total Population	Negative
Population Density	Positive
White Black Asian Hispanic	Positive
Bachelor’s Degree or Higher	Negative

Regression Model

Linear regression

Lasso Regression

Random Forest Regression & XGBoost

Gradient Boosting Regressor

Random Forest Regression & XGBoost

```
from sklearn.ensemble import RandomForestRegressor

#create regressor object
model = RandomForestRegressor(n_estimators= 100, random_state= 0)

#fit the regressor with x and y data
model.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
y_pred= model.predict(X_test)
```

```
print(model.score(X_train, y_train))

print(model.score(X_test, y_test))
```

```
0.953771167771093
0.7887603072085864
```

```
import xgboost as xg
from sklearn.metrics import mean_squared_error as MSE
```

```
#Instantiation

xgb_r = xg.XGBRegressor(objective= 'reg:linear', n_estimators= 100, seed = 123)

xgb_r.fit(X_train, y_train)

y_pred_xgb = xgb_r.predict(X_test)

print(xgb_r.score(X_train, y_train))
print(xgb_r.score(X_test, y_test))
```

```
0.9495909439186249
0.756339597268104
```

Regression Model

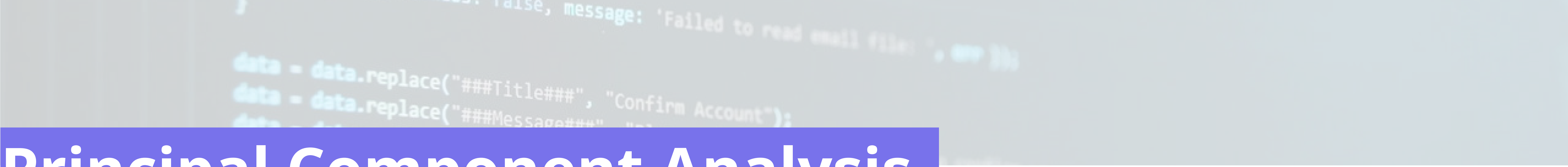
- Linear regression
- Lasso Regression
- Random Forest Regression & XGBoost
- **Gradient Boosting Regressor**

Gradient Boosting Regressor

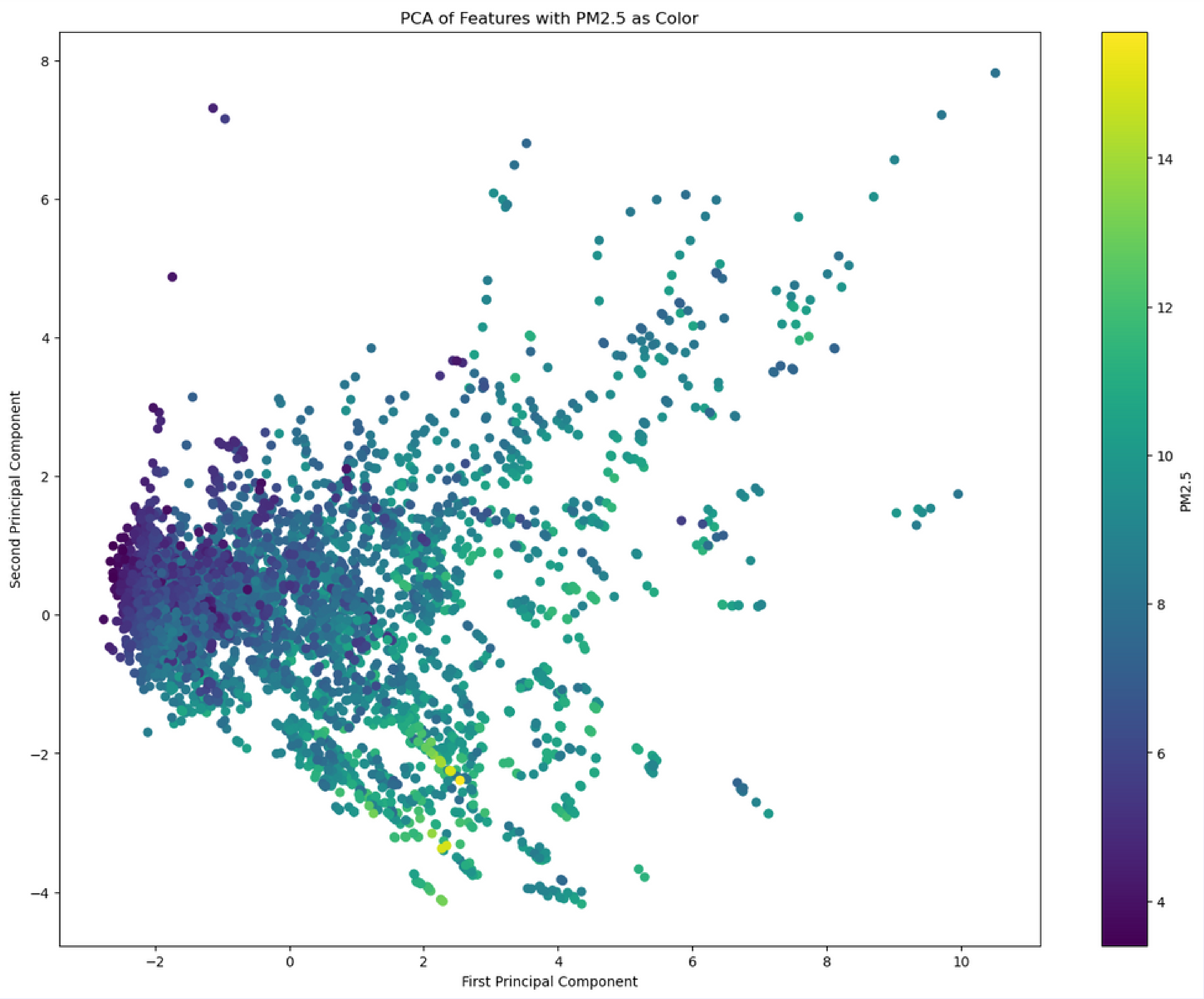
```
from sklearn.ensemble import GradientBoostingRegressor
```

```
# Instantiate Gradient Boosting Regressor
gbr = GradientBoostingRegressor(n_estimators = 100, max_depth = 1)

# Fit to training set
gbr.fit(X_train, y_train)

# Predict on test set
pred_y = gbr.predict(X_test)

# test set RMSE
test_rmse = MSE(y_test, pred_y) ** (1 / 2)

print(gbr.score(X_train, y_train))
print(gbr.score(X_test, y_test))
```

```
0.5146372829636321
0.4992482908383168
```



Principal Component Analysis



Hard to see the clear pattern!

Pipeline

With PCA

```python
for i,model in enumerate(pipelines):
    print("{} Test Accuracy: {}".format(pipe_dict[i], model.score(X_test,y_test)))
```

```
Linear Regression Test Accuracy: 0.3524967527659876
Decision Tree Regressor Test Accuracy: 0.2773010752709888
Random Forest Regressor Test Accuracy: 0.5729584606990209
Support Vector Regressor Test Accuracy: 0.39831525573017257
Gradient Boosting Regressor Test Accuracy: 0.4837166608453629
```

Without PCA

```python
# Pipeline for: decision tree, random forest regressor, SVC, GradientBoosting Regerssor
for pipe in pipelines:
    pipe.fit(X_train, y_train)
for i,model in enumerate(pipelines):
    print("{} Test Accuracy: {}".format(pipe_dict[i], model.score(X_test,y_test)))
```

```
Linear Regression Test Accuracy: 0.39919124655557614
Decision Tree Regressor Test Accuracy: 0.6195287179347277
Random Forest Regressor Test Accuracy: 0.7828656586754518
Support Vector Regressor Test Accuracy: 0.5185248684004499
Gradient Boosting Regressor Test Accuracy: 0.6397047360077701
```

GridSearchCV

```
In [83]: # Specify hyperparameters for the RandomForestRegressor. Testing with different values for hyperparameters
parameters = {
    'regressor__max_features': ['sqrt', 'log2', None],
    'regressor__max_leaf_nodes': [10, 100, 200, None],
    'regressor__n_estimators': [10, 100, 500, 1000],
    'regressor__bootstrap': [True, False],
    'scaler': [StandardScaler(), MinMaxScaler(), Normalizer(), MaxAbsScaler()]

}
```

```
In [84]: grid = GridSearchCV(pipeline_rf, parameters, cv=2).fit(X_train, y_train)
```

```
In [85]: print('Training set score: ' + str(grid.score(X_train, y_train)))
print('Test set score: ' + str(grid.score(X_test, y_test)))
```

```
Training set score: 0.9547499011962798
Test set score: 0.7931548759013509
```

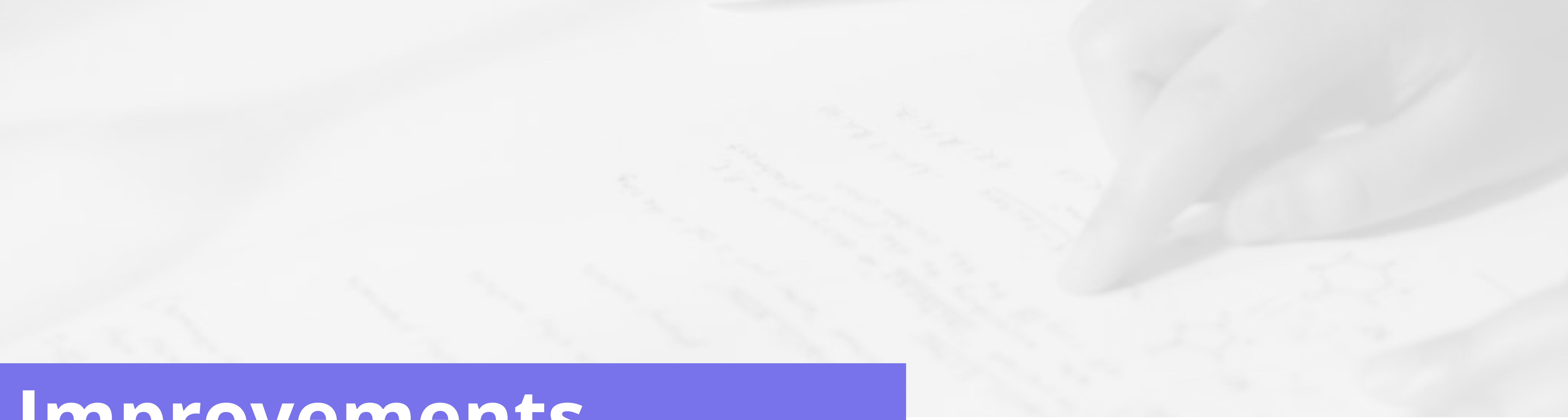
```
In [86]: #Access the best set of parameters

best_params= grid.best_params_
print(best_params)

best_pipe= grid.best_estimator_
print(best_pipe)
```

```
{'regressor__bootstrap': True, 'regressor__max_features': 'sqrt', 'regressor__max_leaf_nodes': None, 'regressor__n_estimators': 500, 'scaler': StandardScaler()}
Pipeline(steps=[('scaler', StandardScaler()), ('selector', VarianceThreshold()),
                ('regressor',
                 RandomForestRegressor(max_features='sqrt', n_estimators=500))])
```

Conclusion

In Summary, we used ZCTA, ACS, and traffic data to analyze, compare, and correlate with the goal of establishing a prediction system on how pollution (PM2.5) relates to traffic and demographic information.

We found that Random Forest Regression hyperparameters provided the best values



Improvements

To further improve this experiment, we could add more parameters to the gridsearch

We acknowledge several limitations presented by this experiment

Questions?