Predictive Models of Methane Leak Rate

January 8, 2023

1 ENERES 131 Connector Assistant Research Project

Author: Richard Wang

1.1 Motivation

This research is originally conducted to see if the course staffs may improve one of the homework in the course. The original homework only has data from **CalEnviroScreen**. We would like to add another dataset and merge it with the CalEnviroScreen dataset, and use a feature from our new dataset as the target variable.

In the end, this new interation of the homework was not implemented in the course, but I continued on the project. My goal in this research is to **find the best set of features that predicts methane leakage rate, so in the future when people try to prevent methane leakage, they can use these set of features as a starting point.**

1.2 Background on the new dataset - Methane Data

The data is provided by the **Environmental Defense Fund**. The portion we will be using contains only methane leakages collected in **Los Angeles**. - The measurements are the **methane leak rates** in a specific time span. - The measurements are measured in levels: *Low, Medium, High*. The feature is called **Leak_Cat**. - The measurements are measured **Per Geo Coordinate**, meaning every measurement is measured in a different geo coordinate - The time span is indicated in features **FIRST_D** and **LAST_D**

1.3 Research Question

Which features provide the most most predictive power for methane leakage rate?

1.3.1 Import Libraries

[ ]:
```
# Run this cell to import the packages we will need
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import geopandas as gpd
```

```
import requests
import seaborn as sns
import xgboost as xgb

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import SVR
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
%matplotlib inline
```

1.4 Part 1: EDA and Data Cleaning

1.4.1 Datasets used

- **LA_Shp**: Methane leak rate data provided by the EDF
- **ces20**: CalEnviroScreen environmental data

[2]:
```
LA_shp = gpd.read_file('LosAngeles/Los_Angeles.shp')
ces20 = pd.read_csv('CES20/ces20.csv')
```

[3]:
```
LA_shp.head()
```

[3]:
```
   FID_   FIRST_D    LAST_D        Lat         Lon Leak_Cat  \
0   0.0  20150206  20150416  33.778466 -117.859590      Low
1   1.0  20150322  20150406  33.780237 -117.863569      Low
2   2.0  20150314  20150415  33.780511 -117.871238      Low
3   3.0  20150211  20150406  33.780921 -117.841545      Low
4   4.0  20150313  20150314  33.781181 -117.867509      Low

                    geometry
0  POINT (-117.85959 33.77847)
1  POINT (-117.86357 33.78024)
2  POINT (-117.87124 33.78051)
3  POINT (-117.84154 33.78092)
4  POINT (-117.86751 33.78118)
```

[4]:
```
ces20.head()
```

[4]:
```
   Census Tract  Total Population California County    ZIP    City  \
0    6019001100              3174            Fresno  93706  Fresno
1    6019000300              3609            Fresno  93706  Fresno
```

```
2    6019000200              3167          Fresno   93706  Fresno
3    6019001500              2206          Fresno   93725  Fresno
4    6019000600              6161          Fresno   93721  Fresno

    Longitude   Latitude Click for interactive map  CES 2.0 Score  \
0 -119.781696  36.709695             Click for map          89.22
1 -119.801035  36.726462             Click for map          83.71
2 -119.805504  36.735491             Click for map          83.47
3 -119.717843  36.681600             Click for map          83.08
4 -119.793357  36.743063             Click for map          82.95

  CES 2.0 \nPercentile Range  … Education Pctl  Linguistic Isolation  \
0   96-100% (highest scores)  …          95.60                  21.6
1   96-100% (highest scores)  …          91.05                  18.3
2   96-100% (highest scores)  …          93.95                  16.2
3   96-100% (highest scores)  …          90.48                  19.5
4   96-100% (highest scores)  …          90.09                  16.9

   Linguistic Isolation Pctl    Poverty  Poverty Pctl  Unemployment  \
0                      83.66  77.500865         97.78         19.30
1                      78.34  81.204032         98.93           NaN
2                      74.04  86.828423         99.66         25.27
3                      80.68  62.746088         88.32         18.30
4                      75.49  88.680993         99.79         26.69

   Unemployment Pctl  Pop. Char.   Pop. Char. Score  Pop. Char. Pctl
0              92.05   90.072268           9.360658            99.60
1                NaN   92.323243           9.594592            99.90
2              98.14   91.499039           9.508958            99.80
3              89.60   83.474281           8.674967            97.74
4              98.61   92.246260           9.586590            99.89

[5 rows x 55 columns]
```

1.4.2 Census Function

Although both datasets has **geo-coordinates** as features, merging on them results in an empty dataset because geo-coordinates are too specific, resulting in no matches between the two datasets.

So I decided to convert geo-coordinates to their belonging census tracts, and **merge the datasets on matching census tracts**. I did so using an API provided by the **Federal Communications Commission**

[5]:
```
#Function to convert geo-coordinates to Census Tract
def Census(lat, lon):
  url = "https://geo.fcc.gov/api/census/area"
  r = requests.get(url, params = {'lat': lat, 'lon': lon})
  census = int(r.json()['results'][0]['block_fips'][1:11])
```

```
    return census
```

1.4.3 Data cleaning and merge the two datasets

[6]:
```
#Clean out nulls
LA_shp = LA_shp[LA_shp['Lat'] != 0]

#Add Census Tract column to LA_shp
LA_shp['Census Tract'] = LA_shp.apply(lambda df: Census(df.Lat, df.Lon), axis =␣
 ↪1)

#Merged data: CalEnviroScreen merged with LA Methane Leak on Census Tract
merged = pd.merge(ces20, LA_shp, how = 'inner', left_on = 'Census Tract',␣
 ↪right_on = 'Census Tract')

#Drop unused features and duplicates
new_merged = merged.drop(['FID_', 'FIRST_D', 'LAST_D', 'Lat', 'Lon',␣
 ↪'geometry'], axis = 1)
new_merged = new_merged.drop_duplicates()
```

Because every census tract may have multiple methane leak rate measurements, I grouped by census tracts, and used different aggregate functions to create the target variable: - **Leak_Rate_Average**: the average of leak rates in the census tract. Low = 0, Medium = 1, High = 2

[ ]:
```
#group by census tract and creating new target variables
census_leak = new_merged[['Census Tract', 'Leak_Cat']]

census_leak['Leak_Rate'] = census_leak['Leak_Cat'].map({'Low': 0, 'Medium': 1,␣
 ↪'High': 2})

census_leak = census_leak.groupby('Census Tract').agg({
    'Leak_Cat': lambda x: list(x),
    'Leak_Rate': 'mean'
})

census_leak = census_leak.rename(columns = {'Leak_Rate': 'Leak_Rate_Average'})
census_leak['Leakages'] = census_leak['Leak_Cat']
new_merged = new_merged.drop(['Click for interactive map', 'Hyperlink'], axis =␣
 ↪1)
```

Merge the new target variables with the cleaned CalEnviroScreen data.

[8]:
```
new_merged = pd.merge(new_merged, census_leak, how = 'left', left_on = 'Census␣
 ↪Tract', right_on = 'Census Tract')
new_merged = new_merged.dropna()
```

1.5 Part 2: Modeling

1.5.1 Feature selection

For feature selection, I found the **linear correlation** between every feature and our target variable, and selected the features that has a **correlation > 0.1 or < -0.1**. After dropping the duplicating features, I am left with **14** features for the target variable.

[9]:
```
new_merged = new_merged.drop(columns = ['Census Tract', 'California County', 'ZIP'])
corr = new_merged.corr()
```

[10]:
```
LeakAverageCorr = corr['Leak_Rate_Average']

LeakAverageFeatures = LeakAverageCorr[(LeakAverageCorr > 0.1) | (LeakAverageCorr < -0.1)]
LeakAverageFeatures = LeakAverageFeatures.drop(labels = ['Diesel PM', 'Imp. Water Bodies Pctl', 'Asthma Pctl', 'Low Birth Weight Pctl', 'Poverty Pctl', 'Unemployment Pctl', 'Pop. Char. ', 'Pop. Char. Pctl', 'Leak_Rate_Average'])
```

[11]:
```
'''new_merged = new_merged.dropna()
X = new_merged.loc[:, (new_merged.columns != 'Has_Leakage') &
                   (new_merged.columns != 'California County') &
                  (new_merged.columns != 'City') &
                  (new_merged.columns != 'Highest_Leakage') &
                  (new_merged.columns != 'Leakages') &
                  (new_merged.columns != 'Num_Leakages') &
                  (new_merged.columns != 'CES 2.0 \nPercentile Range')]
X = X.set_index('Census Tract')'''
X = new_merged[LeakAverageFeatures.index]
y = new_merged['Leak_Rate_Average']
y = y.astype('float')
```

[12]:
```
#train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

For modeling, I used **three** models: **Random Forest, XGBoost, and a Multi-layer Perceptron Neural Net.** For each model, I used **GridSearchCV** from sklearn for hyperparameter tuning and cross validation. I'm doing a **5-fold** CV.

1.5.2 Random Forest

[13]:
```
forest = RandomForestRegressor(random_state = 42)

param_grid = {
```

```
    'n_estimators': [5, 20, 50, 100],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 6, 10],
    'min_samples_leaf': [1, 3, 4],
    'bootstrap': [True, False]
}

randomForest = GridSearchCV(estimator = forest, param_grid = param_grid, cv =␣
 ↪5, verbose = False)
randomForest.fit(X_train, y_train)
randomForestPred = randomForest.best_estimator_.predict(X_test)
```

1.5.3 XGBoost

[14]:
```
xgb_model = xgb.XGBRFRegressor(random_state = 42)

param_grid = {
    'learning_rate': [ 0.01, 0.05],
    'max_depth': [5, 10, 15],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': np.arange(0.5, 1.0, 0.1),
    'n_estimators': [50, 100, 150],
}

xgboost = GridSearchCV(estimator = xgb_model, param_grid = param_grid, cv = 5,␣
 ↪verbose = False)
xgboost.fit(X_train, y_train)
xgboostPred = xgboost.best_estimator_.predict(X_test)
```

1.5.4 Neural network - Multi-layer Perceptron

[ ]:
```
mlp = MLPRegressor(random_state = 42)

param_grid = {
    'alpha': [0.0001, 0.001, 0.1],
    'activation': ['relu', 'tanh', 'logistic'],
    'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,1)]
}

mlpModel = GridSearchCV(estimator = mlp, param_grid = param_grid, cv = 5,␣
 ↪verbose = False)
mlpModel.fit(X_train, y_train)
mlpPred = mlpModel.best_estimator_.predict(X_test)
```

1.6 Part 3: Result

I used **three** methods for measuring the error for the model predictions: **Mean Squared Error, Mean Absolute Error, and R2 score**.

[16]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

#mse
randomForestMse = mean_squared_error(y_test, randomForestPred)
xgbMse = mean_squared_error(y_test, xgboostPred)
mlpMse = mean_squared_error(y_test, mlpPred)


#mae
randomForestMae = mean_absolute_error(y_test, randomForestPred)
xgbMae = mean_absolute_error(y_test, xgboostPred)
mlpMae = mean_absolute_error(y_test, mlpPred)


#r2
randomForestR2 = r2_score(y_test, randomForestPred)
xgbR2 = r2_score(y_test, xgboostPred)
mlpR2 = r2_score(y_test, mlpPred)
```

[17]:
```
print('Random Forest MSE: ', randomForestMse)
print('XGBoost MSE: ', xgbMse)
print('MLP MSE: ', mlpMse)
print()
print('Random Forest MAE: ', randomForestMae)
print('XGBoost MAE: ', xgbMae)
print('MLP MAE: ', mlpMae)
print()
print('Random Forest R2: ', randomForestR2)
print('XGBoost R2: ', xgbR2)
print('MLP R2: ', mlpR2)
```

```
Random Forest MSE:  0.08370075707959367
XGBoost MSE:  0.10499091561245155
MLP MSE:  0.11442918253792207

Random Forest MAE:  0.25114127492488836
XGBoost MAE:  0.2398215596164976
MLP MAE:  0.26718253781113593

Random Forest R2:  0.2626809713438043
XGBoost R2:  0.0751362040431236
MLP R2:  -0.008005383255403231
```

2 Conclusion

This research project's main purpose is to first: **see if predicting methane leak rate with environmental data from CalEnviroScreen is possible**, and if so, **what are the best features and models for it?** With the results from the three models I've tested, the three models performed similarly in predicting methane leak rates.

2.0.1 Most Predictive Features

These are the 14 features and their correlation with leakage rate, ranked from **most predictive to least**: - Unemployment: **.268081** - Pop. Char. Score: **0.228498** - Asthma: **0.211940** - Poverty: **0.187427** - Imp. Water Bodies: **-0.185010** - CES 2.0 Score: **0.173223** - Low Birth Weight: **0.169838** - Education Pctl: **0.145919** - Traffic Pctl: **-0.144668** - Age Pctl: **-0.128569** - Pesticides Pctl: **-0.122472** - Cleanup Sites Pctl: **0.121789** - Linguistic Isolation Pctl: **0.120849** - Diesel PM Pctl: **0.103990**

2.0.2 Potential issues and flaws

- **Size of data**: the biggest issue is I am working with a dataset that has around 60 measurements due to the mismatches in the merging process, so the results is potentially very biased
- **Nature of the datasets**: the two datasets are measured different, one measured **per census**, one measured **per coordinate**. Although the API resolved the issue, it largely reduced the number of measurements I can work with

2.0.3 Future improvements

Add data: try a dataset that is measured in **per census**, and also cover more California states other than Los Angeles. A good place to start is to find data on the best features for predicting leak rates, as listed above. The data with those features that is also measured in per census will work the best.