**Team Project – team 11**

201835100 장현우 (Jang Hyeon Woo)

201835100 최현준 (Choi Hyun jun)

201935100 이승민 (Lee Seung Min)

202135000 정성수 (Jung Seong Soo)

**<Source Code>**

**<preprocessing\_seoul\_data.py>**

import pandas as pd # for data analysis  
from matplotlib import font\_manager, rc # for plotting Korean font  
  
font\_path = '/Library/Fonts/Arial Unicode.ttf'  
  
# 경기도 성남시\_주정차 위반 단속 위치 현황  
# https://www.data.go.kr/data/15037104/fileData.do  
#  
# 서울특별시 강동구\_동별 불법주정차 단속현황  
# https://www.data.go.kr/data/15081659/fileData.do  
#  
# 서울특별시 강서구\_불법주정차 단속 현황  
# https://www.data.go.kr/data/15083768/fileData.do  
#  
# 서울특별시 구로구\_주정차단속현황  
# https://www.data.go.kr/data/15034492/fileData.do  
#  
# 서울특별시 서초구\_주정차 단속 현황  
# https://www.data.go.kr/data/15087185/fileData.do  
#  
# 서울특별시 성북구\_불법주정차 동별 데이터  
# https://www.data.go.kr/data/15113658/fileData.do  
#  
# 서울특별시 송파구\_주정차단속건수정보  
# https://www.data.go.kr/data/15048835/fileData.do  
#  
# 서울특별시 영등포구\_주정차단속현황  
# https://www.data.go.kr/data/15034483/fileData.do  
#  
# 서울특별시 용산구\_불법주정차단속현황  
# https://www.data.go.kr/data/15084175/fileData.do  
#  
# 서울특별시 종로구\_불법주정차 통계  
# https://www.data.go.kr/data/15100293/fileData.do  
#  
# 서울특별시\_강남구\_불법주정차단속현황  
# https://www.data.go.kr/data/15048827/fileData.do  
#  
# 서울특별시\_서대문구\_주정차 단속 현황  
# https://www.data.go.kr/data/15034465/fileData.do  
#  
# 소상공인시장진흥공단\_상가(상권)정보  
# https://www.data.go.kr/data/15083033/fileData.do  
  
  
  
  
# font setting  
font\_name = font\_manager.FontProperties(fname=font\_path).get\_name()  
  
rc('font', family=font\_name)  
  
  
  
# ---------------------강동구 불러오기---------------------  
print("---------------------강동구 불러오기---------------------")  
address = "추가데이터/서울특별시 강동구\_동별 불법주정차 단속현황\_20201231.csv"  
df1 = pd.read\_csv(address, encoding = 'cp949')  
print(df1.describe())  
  
# remain = ['단속동', '단속건수']  
df1 = df1.drop(['기준일자'], axis=1)  
print(df1)  
  
# ---------------------서초구 불러오기---------------------  
print("---------------------서초구 불러오기---------------------")  
address = "추가데이터/서울특별시 서초구\_주정차 단속 현황\_20210831.csv"  
df2 = pd.read\_csv(address, encoding = 'cp949')  
print(df2.describe())  
  
# '날짜' column convert to datetime  
df2['단속일시'] = pd.to\_datetime(df2['단속일시'])  
  
# add '연도' column  
df2['연도'] = df2['단속일시'].dt.year  
  
# preprocessing for 단속동 column ex 서초1동 -> 서초동  
df2['단속동'] = df2['단속동'].str.extract(r'(\w+[동])')  
  
  
# calculate mean of 단속건수 by 연도, 단속동  
df\_grouped = df2.groupby(['연도', '단속동']).size().reset\_index(name='단속건수')  
  
df2 = df\_grouped.groupby('단속동')['단속건수'].mean().reset\_index(name='단속건수')  
print(df2)  
  
  
# ---------------------송파구 불러오기---------------------  
print("---------------------송파구 불러오기---------------------")  
address = "추가데이터/서울특별시 송파구\_주정차단속건수정보\_20200313..csv"  
df3 = pd.read\_csv(address, encoding = 'cp949')  
print(df3.describe())  
  
# preprocessing for 단속동 column  
df3['단속동'] = df3['단속동'].str.extract(r'(\w+[동])')  
  
# calculate mean of 단속건수 by 단속동  
df3 = df3.groupby('단속동')['단속건수'].mean().reset\_index(name='단속건수')  
  
print(df3)  
  
# ---------------------강남구 불러오기---------------------  
print("---------------------강남구 불러오기---------------------")  
address = "추가데이터/서울특별시\_강남구\_불법주정차단속현황\_20220207.csv"  
df4 = pd.read\_csv(address, encoding = 'cp949')  
print(df4.describe())  
  
# trim space in 동명 column  
df4['동명'] = df4['동명'].str.replace(' ', '')  
  
# calculate mean of 단속건수 by 동명  
df4 = df4.groupby('동명')['부과건수'].mean().reset\_index(name='단속건수')  
  
# rename column name  
df4.rename(columns={'동명': '단속동'}, inplace=True)  
print(df4)  
  
# ---------------------강서구 불러오기---------------------  
print("---------------------강서구 불러오기---------------------")  
address = "추가데이터/서울특별시 강서구\_불법주정차 단속 현황\_20201231.csv"  
df5 = pd.read\_csv(address, encoding = 'cp949')  
print(df5.describe())  
  
# calculate mean of 단속건수 by 행정동명  
df5 = df5.groupby('행정동명')[' 단속건수 '].mean().reset\_index(name=' 단속건수 ')  
  
# rename column name  
df5.rename(columns={'행정동명': '단속동'}, inplace=True)  
df5.rename(columns={' 단속건수 ': '단속건수'}, inplace=True)  
  
print(df5)  
  
# ---------------------구로구 불러오기---------------------  
print("---------------------구로구 불러오기---------------------")  
address = "추가데이터/서울특별시 구로구\_주정차단속현황\_20230127.csv"  
df6 = pd.read\_csv(address, encoding = 'cp949')  
print(df6.describe())  
  
# calculate mean of 단속건수 by 단속동  
df6 = df6.groupby('단속동')['단속건수'].mean().reset\_index(name='단속건수')  
  
print(df6)  
  
# ---------------------영등포구 불러오기---------------------  
print("---------------------영등포구 불러오기---------------------")  
address = "추가데이터/서울특별시 영등포구\_주정차단속현황\_20230504.csv"  
df7 = pd.read\_csv(address, encoding = 'cp949')  
print(df7.describe())  
  
# preprocessing for 단속동 column  
df7['단속동'] = df7['단속동'].str.extract(r'(\w+[동])')  
  
# preprocessing for 단속년도 column 2022년 -> 2022  
df7['단속년도'] = df7['단속년도'].str.extract(r'(\d{4})')  
  
# calculate mean of 단속건수 by 단속동  
df7 = df7.groupby('단속동')['단속건수'].mean().reset\_index(name='단속건수')  
  
print(df7)  
  
# ---------------------용산구 불러오기---------------------  
print("---------------------용산구 불러오기---------------------")  
address = "추가데이터/서울특별시 용산구\_불법주정차단속현황\_10\_25\_2021.csv"  
df8 = pd.read\_csv(address, encoding = 'cp949')  
print(df8.describe())  
  
# convert 단속일시 column to datetime type  
df8['단속일시'] = pd.to\_datetime(df8['단속일시'])  
  
# add 연도 column  
df8['연도'] = df8['단속일시'].dt.year  
  
# preprocessing for 단속동 column  
df8['단속동'] = df8['단속동'].str.extract(r'(\w+[동])')  
  
# calculate mean of 단속건수 by 단속동  
df\_grouped = df8.groupby(['연도', '단속동']).size().reset\_index(name='단속건수')  
df8 = df\_grouped.groupby('단속동')['단속건수'].mean().reset\_index(name='단속건수')  
  
print(df8)  
  
# ---------------------서대문구 불러오기---------------------  
print("---------------------서대문구 불러오기---------------------")  
address = "추가데이터/서울특별시\_서대문구\_주정차 단속 현황\_20220809.csv"  
df9 = pd.read\_csv(address, encoding = 'cp949')  
print(df9.describe())  
  
# preprocessing for 단속동 column  
df9['단속동'] = df9['단속동'].str.extract(r'(\w+[동])')  
  
# calculate mean of 단속건수 by 단속동  
df9 = df9.groupby(['단속동']).size().reset\_index(name='단속건수')  
  
print(df9)  
  
# ---------------------종로구 불러오기---------------------  
print("---------------------종로구 불러오기---------------------")  
address = "추가데이터/서울특별시 종로구\_불법주정차 통계\_20211231.csv"  
df10 = pd.read\_csv(address, encoding = 'cp949')  
print(df10.describe())  
print(df10.info)  
  
# drop 기간 column  
df10 = df10.drop(['기간'], axis=1)  
  
# rename column name  
df10.rename(columns={'행정동': '단속동'}, inplace=True)  
df10.rename(columns={'단속건': '단속건수'}, inplace=True)  
  
print(df10)  
  
# ---------------------성북구 불러오기---------------------  
print("---------------------성북구 불러오기---------------------")  
address = "추가데이터/서울특별시 성북구\_불법주정차 동별 데이터\_20230504.csv"  
df11 = pd.read\_csv(address, encoding = 'cp949')  
print(df11.describe())  
  
# preprocessing for 단속동 column  
df11['단속동'] = df11['단속동'].str.extract(r'(\w+[동])')  
  
# calculate sum of 단속건수 by 단속동  
df11 = df11.groupby('단속동')['단속건수'].sum().reset\_index(name='단속건수')  
  
print(df11)  
  
# merge all dataframes  
df = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11], axis=0, ignore\_index=True)  
df = df.groupby('단속동')['단속건수'].sum().reset\_index(name='단속건수')  
  
  
print("---------------------결과 불러오기---------------------")  
print(df)  
print(df.describe())  
  
# ---------------------서울 인구수 데이터 불러오기---------------------  
print("---------------------서울 인구수 데이터 불러오기---------------------")  
address = "추가데이터/인구밀도\_20230603023951.csv"  
df\_sum = pd.read\_csv(address, encoding = 'utf-8')  
print(df\_sum)  
  
# create new dataframe with only '동별(3)' and '인구 (명)' columns  
df\_sum = df\_sum[['동별(3)', '인구 (명)']]  
  
# rename column name  
df\_sum.rename(columns={'동별(3)': '단속동'}, inplace=True)  
  
# calculate sum of 인구 (명) by 단속동  
df\_sum = df\_sum.groupby('단속동')['인구 (명)'].sum().reset\_index(name='인구 (명)')  
  
print(df\_sum)  
  
  
# ---------------------서울시\_동별\_단속현황.xlsx 파일로 저장---------------------  
print("---------------------서울시\_동별\_단속현황.xlsx 파일로 저장---------------------")  
# df와 df\_sum를 하나의 데이터 프레임으로  
df\_preprocessing = pd.merge(df, df\_sum, on='단속동', how='inner')  
print(df\_preprocessing)  
  
df\_preprocessing.to\_excel('서울시\_동별\_단속현황.xlsx', index = False)  
  
# # '인구' column명 변경  
# df\_preprocessing.rename(columns={'인구 (명)': '인구'}, inplace=True)  
#  
# # 단속동이 청담동, 삼성동, 신사동 row는 삭제  
# df\_preprocessing = df\_preprocessing.drop(df\_preprocessing[df\_preprocessing['단속동'] == '청담동'].index)  
# df\_preprocessing = df\_preprocessing.drop(df\_preprocessing[df\_preprocessing['단속동'] == '삼성동'].index)  
# df\_preprocessing = df\_preprocessing.drop(df\_preprocessing[df\_preprocessing['단속동'] == '신사동'].index)

**<Output>**

|  |
| --- |
| Print dataset and after preprocessing for merging |
| 텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print merged dataset – 단속동, 단속건수 |
| 텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print dataset – 서울 인구수 데이터 |
| 텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진  자동 생성된 설명 |

**<final.py>**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split, KFold  
from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder, OneHotEncoder  
from sklearn.tree import DecisionTreeClassifier, export\_graphviz  
from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score, f1\_score, classification\_report  
from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, mean\_absolute\_percentage\_error, accuracy\_score  
from matplotlib import font\_manager, rc  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.linear\_model import Ridge, Lasso, LinearRegression  
from xgboost import XGBClassifier  
  
  
font\_path = '/Library/Fonts/Arial Unicode.ttf'  
font\_name = font\_manager.FontProperties(fname=font\_path).get\_name()  
rc('font', family=font\_name)  
  
import warnings  
warnings.filterwarnings("ignore")  
  
# KFold evaluation  
# input : X, y, k, mode  
# output : None(void)  
# description : KFold evaluation about each classification model and print average accuracy  
def evaluate\_Kfold(X, y, k, mode):  
 kf = KFold(n\_splits=k, shuffle=True, random\_state=42)  
  
 accuracy\_scores = []  
  
 for train\_index, test\_index in kf.split(X):  
 X\_train, X\_test = X.iloc[train\_index], X.iloc[test\_index]  
 y\_train, y\_test = y.iloc[train\_index], y.iloc[test\_index]  
  
 model = None  
  
 if mode == 'Decision Tree':  
 model = DecisionTreeClassifier(max\_depth=5, random\_state=42)  
 model.fit(X\_train, y\_train)  
 elif mode == 'KNN':  
 model = KNeighborsClassifier(n\_neighbors=5)  
 model.fit(X\_train, y\_train)  
 elif mode == 'Random Forest':  
 model = RandomForestClassifier()  
 model.fit(X\_train, y\_train)  
 elif mode == 'XGBoost':  
 model = XGBClassifier(random\_state=42)  
 model.fit(X\_train, y\_train)  
  
 y\_pred = model.predict(X\_test)  
  
 accuracy = accuracy\_score(y\_test, y\_pred)  
  
 accuracy\_scores.append(accuracy)  
  
 average\_accuracy = sum(accuracy\_scores) / k # 평균 정확도 계산  
  
 print(f"{mode} Average accuracy: {average\_accuracy:.2f}")  
  
  
# evaluation\_classification  
# input : y\_true, y\_pred  
# output : None(void)  
# description : print accuracy, confusion matrix, precision, recall, f1 score  
def evaluation\_classification(y\_true, y\_pred):  
 # calculate accuracy  
 accuracy = accuracy\_score(y\_true, y\_pred)  
  
 # calculate confusion matrix  
 confusion\_mtx = confusion\_matrix(y\_true, y\_pred)  
  
 # calculate precision  
 precision = precision\_score(y\_true, y\_pred, average='macro') # multi-class -> average='macro'  
  
 # calculate recall  
 recall = recall\_score(y\_true, y\_pred, average='macro') # multi-class -> average='macro'  
  
 # calculate f1 score  
 f1 = f1\_score(y\_true, y\_pred, average='macro') # multi-class -> average='macro'  
  
 # print  
 print(f"Accuracy: {accuracy:.2f}")  
 print("Confusion Matrix:")  
 print(confusion\_mtx)  
 print(f"Precision: {precision:.2f}")  
 print(f"Recall: {recall:.2f}")  
 print(f"F1 Score: {f1:.2f}")  
  
print("---------- 단속위치 데이터 ---------")  
address = "경기도 성남시\_주정차 위반 단속 위치 현황\_20220927.csv"  
df1 = pd.read\_csv(address, encoding = 'cp949')  
  
  
words\_to\_check = ["창곡", "양지", "은행", "산성", "단대", "금광", "상대원", "갈현", "도촌", "여수", "하대원", "성남", "수진", "신흥", "복정", "태평", "신촌", "오야", "심곡", "둔전", "고등", "시흥", "사송", "상적", "금토", "야탑", "이매", "율", "서현", "분당", "수내", "정자", "구미", "금곡", "동원", "궁내", "백현", "판교", "삼평", "하산운", "대장", "운중", "석운"]  
  
df1['단속장소'] = df1['단속장소'].str.extract(r'(\w+[동])')  
  
  
# 시군명, 관리기관명, 단속일시정보, 단속방법 column을 제거  
df1 = df1.drop(['시군명', '관리기관명', '단속일시정보', '단속방법'], axis = 1).reset\_index()  
df1 = df1.drop(['index'], axis = 1)  
  
  
#------------------- words\_to\_check 리스트 안에 있는 단어들이 있는 row들을 통일시킴 -------------------  
pattern = r'\b(\w+동)\b'  
  
mask = df1['단속장소'].str.contains('|'.join(words\_to\_check), case=False, na=False)  
df1.loc[mask, '단속장소'] = df1.loc[mask, '단속장소'].str.extract(pattern, expand=False).fillna(df1.loc[mask, '단속장소'])  
  
  
#------------------- words\_to\_check 리스트 안에 있는 단어들이 있는 row들을 해당 단어가 포함된 부분만으로 대체 ---------------  
df1 = df1.dropna(how='any').reset\_index()  
df1 = df1.drop(['index'], axis = 1)  
  
# calculate the number of rows in the dataframe  
df\_grouped = df1.groupby(['집계년도', '단속장소']).size().reset\_index(name='단속횟수')  
  
# calculate the mean of the number of rows in the dataframe  
df\_counts = df\_grouped.groupby('단속장소')['단속횟수'].mean().reset\_index(name='단속횟수')  
  
# filter out the rows that have less than 10 counts <Outlier>  
df\_counts = df\_counts[df\_counts['단속횟수'] > 10]  
  
# sort the rows by the number of counts in descending order  
df\_counts = df\_counts.sort\_values(by=['단속횟수'], axis=0, ascending=False).reset\_index()  
df\_counts = df\_counts.drop(['index'], axis = 1)  
  
# create a new dataframe  
df\_new = pd.DataFrame({'단속장소': df\_counts['단속장소'], '단속횟수': df\_counts['단속횟수']})  
  
# rename the columns  
df\_new = df\_new.rename(columns={'단속장소': '단속동'})  
  
# trim  
df\_new['단속동'] = df\_new['단속동'].str.strip()  
  
df\_new.to\_excel('단속장소\_단속횟수\_final.xlsx', index = False)  
  
print(df\_new)  
  
print("---------- 인구수 ---------")  
address = "경기도 성남시\_인구및세대\_현황\_20230430.csv"  
df\_population = pd.read\_csv(address, encoding = 'utf-8')  
print(df\_population)  
  
# renew column  
df\_population = df\_population[['동', '인구수\_계']]  
  
# rename column  
df\_population = df\_population.rename(columns={'동': '단속동', '인구수\_계': '인구 (명)'})  
  
# trim  
df\_population['단속동'] = df\_population['단속동'].str.strip()  
  
print(df\_population)  
  
print("---------- 성남시 인구수 + 동 합치기 ---------")  
# merge two dataframes  
df\_sn = pd.merge(df\_new, df\_population, how='inner')  
print(df\_sn)  
  
print("---------- 서울시 동별 단속건수 ---------")  
address = "서울시\_동별\_단속현황.xlsx"  
df\_seoul\_data = pd.read\_excel(address)  
print(df\_seoul\_data)  
  
print("---------- 데이터셋 merge ---------")  
  
# rename column  
df\_sn = df\_sn.rename(columns={'단속횟수': '단속건수'})  
  
# merge two dataframes  
df = pd.concat([df\_sn, df\_seoul\_data], axis=0)  
df = df.reset\_index(drop=True)  
  
print(df)  
  
print("---------- 경기도 상권정보 ---------")  
  
address = "소상공인시장진흥공단\_상가(상권)정보\_경기\_202209.csv"  
df\_gyeonggi = pd.read\_csv(address, encoding = 'utf-8')  
print(df\_gyeonggi)  
  
# '상권업종대분류명', '행정동명', '행정동명', '동정보' column  
df\_gyeonggi = df\_gyeonggi[['상권업종대분류명', '행정동명']]  
  
# remove rows that have '중앙동' in '행정동명' column <Unusable Data>  
df\_gyeonggi = df\_gyeonggi[df\_gyeonggi['행정동명'] != '중앙동']  
  
print(df\_gyeonggi)  
  
print("---------- 서울 상권정보 ---------")  
  
address = "소상공인시장진흥공단\_상가(상권)정보\_서울\_202209.csv"  
  
df\_seoul = pd.read\_csv(address, encoding = 'utf-8')  
print(df\_seoul)  
# '상권업종대분류명', '행정동명', '행정동명', '동정보' column  
df\_seoul = df\_seoul[['상권업종대분류명', '행정동명']]  
print(df\_seoul)  
  
print("------------------- 서울 + 경기도 상권정보 합치기 -------------------")  
df\_store = pd.concat([df\_gyeonggi, df\_seoul], axis=0)  
df\_store = df\_store.reset\_index(drop=True)  
  
  
  
# '상권업종대분류명', '행정동명', '행정동명', '동정보' column  
df\_store = df\_store[['상권업종대분류명', '행정동명']]  
  
# 상권업종대분류명 column <one-hot encoding>  
df\_store = pd.get\_dummies(df\_store, columns=['상권업종대분류명'])  
  
# groupby '행정동명' and sum  
df\_combined = df\_store.groupby('행정동명').sum().reset\_index()  
  
# print(df\_combined)  
  
# rename column  
df\_combined = df\_combined.rename(columns={'행정동명': '단속동'})  
  
# merge two dataframes  
df\_merged = pd.merge(df, df\_combined, on='단속동', how='outer')  
  
print("---------- 결측치 제거 ---------")  
# drop rows that have NaN  
print(df\_merged.isnull().sum())  
df\_merged.dropna(inplace=True)  
  
print("------------------- MinMaxScaler -------------------")  
  
# MinMaxScaler  
minmax = MinMaxScaler()  
  
# scale the data  
columns\_to\_scale = ['단속건수', '인구 (명)', '단속동']  
df\_scaled = minmax.fit\_transform(df\_merged.drop(columns\_to\_scale, axis=1))  
  
# reassign the scaled data to its original dataframe  
df\_merged[df\_merged.columns.drop(columns\_to\_scale)] = df\_scaled  
  
print(df\_merged)  
  
  
  
  
print("------------------------------ Data Analysis -------------------------------")  
  
#------------------- 학습시킬 데이터를 제작 -------------------  
minmax = MinMaxScaler()  
X = df\_merged.drop(['단속건수', '단속동'], axis=1)  
X = minmax.fit\_transform(X)  
y = df\_merged['단속건수']  
  
  
#------------------- Split data into training and validation sets -------------------  
X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
ridgeModel = Ridge()  
ridgeModel.fit(X\_train, y\_train)  
  
lassoModel = Lasso()  
lassoModel.fit(X\_train, y\_train)  
  
linearModel = LinearRegression()  
linearModel.fit(X\_train, y\_train)  
  
# #------------------- 사용자로부터 미래에 대한 가정 정보를 받아옴 -------------------  
area = input("예상하고자 하는 지역을 적어주세요 : ")  
rate\_pol, year = input("인구규모 변화율, 예상할 연도를 적어주세요 : ").split()  
rate\_pol = int(rate\_pol)  
year = int(year)  
  
#------------------- 가정을 반영한 새로운 row array를 만듦 -------------------  
# 단속동이 area인 row를 찾음  
predict\_df = df\_merged[df\_merged['단속동'] == area]  
print(predict\_df)  
area\_pol = predict\_df['인구 (명)']  
  
#------------------- 가정을 반영함 -------------------  
for i in range(year - 2023):  
  
 if rate\_pol >= 0:  
 area\_pol = area\_pol + (area\_pol \* (rate\_pol / 100))  
 else :  
 area\_pol = area\_pol - (abs(area\_pol) \* (rate\_pol / 100))  
  
predict\_df['인구 (명)'] = area\_pol  
  
#------------------- X데이터와 함께 scaling하고 그 결과를 저장, scaling으로 인해 변한 자료구조를 다시 dataframe으로 변환함 -------------------  
copy\_X = df\_merged.drop(['단속건수', '단속동'], axis=1)  
minmax.fit(copy\_X)  
  
predict\_df = predict\_df.drop(['단속건수', '단속동'], axis=1)  
  
predict\_df = minmax.transform(predict\_df)  
predict\_df = pd.DataFrame(predict\_df)  
  
#------------------- 사용자의 가정 정보를 토대로 불법주정차 수를 예측 -> Ridge Regression -------------------  
print("\nResult of Regression -> Ridge Regression(user input)")  
y\_pred = ridgeModel.predict(predict\_df.values.reshape(1, -1))  
print(str(year) + "년 " + str(area) + " 지역의 예상 불법주정차 수는 " + str(y\_pred) + " 대 입니다.")  
  
  
#------------------- 이 부분은 test셋으로 split하였던 부분에 대한 예측값 -------------------  
print("\nResult of Regression -> Ridge Regression(test set)")  
y\_pred = ridgeModel.predict(X\_valid)  
  
print(y\_pred)  
  
print("\nErrors of Ridge Regression")  
mse = mean\_squared\_error(y\_valid, y\_pred)  
print(f"Mean Squared Error: {mse}")  
  
# Evaluate the model using mean absolute error (MAE)  
mae = mean\_absolute\_error(y\_valid, y\_pred)  
print(f"Mean Absolute Error: {mae}")  
  
# Evaluate the model using mean absolute percentage error (MAPE)  
mape = mean\_absolute\_percentage\_error(y\_valid, y\_pred)  
print(f"Mean Absolute Percentage Error: {mape}")  
  
print("\nResult of Regression -> Linear Regression(test set)")  
y\_pred = linearModel.predict(X\_valid)  
print(y\_pred)  
  
print("\nErrors of Linear Regression")  
mse = mean\_squared\_error(y\_valid, y\_pred)  
print(f"Mean Squared Error: {mse}")  
  
# Evaluate the model using mean absolute error (MAE)  
mae = mean\_absolute\_error(y\_valid, y\_pred)  
print(f"Mean Absolute Error: {mae}")  
  
# Evaluate the model using mean absolute percentage error (MAPE)  
mape = mean\_absolute\_percentage\_error(y\_valid, y\_pred)  
print(f"Mean Absolute Percentage Error: {mape}")  
  
print("\nResult of Regression -> Lasso Regression(test set)")  
y\_pred = lassoModel.predict(X\_valid)  
  
print(y\_pred)  
  
print("\nErrors of Lasso Regression")  
mse = mean\_squared\_error(y\_valid, y\_pred)  
print(f"Mean Squared Error: {mse}")  
  
# Evaluate the model using mean absolute error (MAE)  
mae = mean\_absolute\_error(y\_valid, y\_pred)  
print(f"Mean Absolute Error: {mae}")  
  
# Evaluate the model using mean absolute percentage error (MAPE)  
mape = mean\_absolute\_percentage\_error(y\_valid, y\_pred)  
print(f"Mean Absolute Percentage Error: {mape}")  
  
  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
print(df\_merged.columns)  
  
df\_plot = df\_merged.copy()  
  
# scale  
columns\_to\_scale = ['인구 (명)', '단속동']  
df\_scaled = minmax.fit\_transform(df\_plot.drop(columns\_to\_scale, axis=1))  
  
# insert scaled columns  
df\_plot[df\_plot.columns.drop(columns\_to\_scale)] = df\_scaled  
  
# visualize  
sns.pairplot(df\_plot, x\_vars=['인구 (명)', '상권업종대분류명\_관광/여가/오락', '상권업종대분류명\_부동산', '상권업종대분류명\_생활서비스', '상권업종대분류명\_소매'],  
 y\_vars='단속건수', kind='reg', height=4)  
plt.show()  
  
sns.pairplot(df\_plot, x\_vars=['상권업종대분류명\_숙박', '상권업종대분류명\_스포츠', '상권업종대분류명\_음식', '상권업종대분류명\_학문/교육'],  
 y\_vars='단속건수', kind='reg', height=4)  
plt.show()  
  
  
X\_candidates = ['인구 (명)', '상권업종대분류명\_관광/여가/오락', '상권업종대분류명\_부동산', '상권업종대분류명\_생활서비스', '상권업종대분류명\_소매', '상권업종대분류명\_숙박', '상권업종대분류명\_스포츠', '상권업종대분류명\_음식', '상권업종대분류명\_학문/교육']  
  
for x in X\_candidates :  
 X\_1 = df\_merged[x].values.reshape(-1, 1)  
 y\_1 = df\_merged['단속건수']  
  
 X\_train\_1, X\_test\_1, y\_train\_1, y\_test\_1 = train\_test\_split(X\_1, y\_1, test\_size=0.2, random\_state=42)  
  
 ridgeModel = Ridge()  
 ridgeModel.fit(X\_train\_1, y\_train\_1)  
  
 y\_pred = ridgeModel.predict(X\_test\_1)  
  
 mse = mean\_squared\_error(y\_test\_1, y\_pred)  
 mae = mean\_absolute\_error(y\_test\_1, y\_pred)  
 mape = mean\_absolute\_percentage\_error(y\_test\_1, y\_pred)  
  
 print("\n" + x + "에 대한 결과")  
 print(f"Mean Squared Error: {mse}")  
 print(f"Mean Absolute Error: {mae}")  
 print(f"Mean Absolute Percentage Error: {mape}")  
  
  
# Evaluate the model using mean squared error (MSE)  
#mse = mean\_squared\_error(y\_valid, y\_pred)  
#print(f"Mean Squared Error: {mse}")  
  
# Step 7: Model Evaluation and Optimization  
  
  
# classification model--------------------------------------------  
print("\n--------------------- Classification Analysis -----------------------")  
# Find mean of '단속횟수'  
count\_mean = df\_merged['단속건수'].mean()  
  
# category function  
def categorize\_count(x):  
 if x < count\_mean:  
 return '적음'  
 else:  
 return '많음'  
  
# add category column  
df\_merged['단속건수\_범주'] = df\_merged['단속건수'].apply(categorize\_count)  
  
# check category  
print(df\_merged['단속건수\_범주'].value\_counts())  
  
print(df\_merged)  
  
print("--------------------- Decision Tree -----------------------")  
  
# Decision Tree  
# Split data into training and test sets  
X = df\_merged.drop(['단속건수', '단속동', '단속건수\_범주'], axis=1)  
y = df\_merged['단속건수\_범주']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Create and fit the Decision Tree classifier  
dt = DecisionTreeClassifier(max\_depth=5, random\_state=42)  
dt.fit(X\_train, y\_train)  
  
# Predict test set labels  
y\_pred = dt.predict(X\_test)  
  
# Evaluate test-set accuracy  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"Decision Tree Test set accuracy: {accuracy:.2f}")  
  
evaluate\_Kfold(X,y,5, "Decision Tree")  
evaluation\_classification(y\_test,y\_pred)  
classification\_report\_result = classification\_report(y\_test, y\_pred)  
print("Classification Report:")  
print(classification\_report\_result)  
  
# visulization  
import graphviz  
from sklearn.tree import export\_graphviz  
  
  
dot\_data = export\_graphviz(dt, out\_file=None, feature\_names=X.columns, class\_names=['적음', '많음'], filled=True, rounded=True)  
  
# visualize  
graph = graphviz.Source(dot\_data)  
  
# save PNG  
graph.format = 'png'  
graph.render(filename='final\_decision\_tree', cleanup=True)  
  
  
print("--------------------- KNN -----------------------")  
from sklearn.neighbors import KNeighborsClassifier  
  
# split data into training and test sets  
# X = df\_merged.drop(['단속횟수', '소속구역명', '단속횟수\_범주', '시장규모', '관공서 수', '교육시설 수', '초중고교 수'], axis=1)  
X = df\_merged.drop(['단속건수', '단속동', '단속건수\_범주'], axis=1)  
y = df\_merged['단속건수\_범주']  
  
# scale the data  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# split the data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
# train KNN classifier  
knn = KNeighborsClassifier(n\_neighbors=5) # K값은 변경 가능  
knn.fit(X\_train, y\_train)  
  
# predict the test set  
y\_pred = knn.predict(X\_test)  
  
# evaluate accuracy  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"KNN Accuracy: {accuracy}")  
evaluation\_classification(y\_test, y\_pred)  
classification\_report\_result = classification\_report(y\_test, y\_pred)  
print("Classification Report:")  
print(classification\_report\_result)  
  
# Define the number of folds  
k = 5  
  
# Perform K-fold cross-validation  
kf = KFold(n\_splits=k, shuffle=True, random\_state=42)  
accuracy\_scores = []  
  
for train\_index, val\_index in kf.split(X\_scaled):  
 X\_train\_k, X\_val = X\_scaled[train\_index], X\_scaled[val\_index]  
 y\_train\_k, y\_val = y[train\_index], y[val\_index]  
  
 knn = KNeighborsClassifier(n\_neighbors=3)  
 knn.fit(X\_train\_k, y\_train\_k)  
 y\_pred = knn.predict(X\_val)  
  
 accuracy = accuracy\_score(y\_val, y\_pred)  
 accuracy\_scores.append(accuracy)  
  
average\_accuracy = sum(accuracy\_scores) / k  
  
print(f"KNN Average accuracy: {average\_accuracy:.2f}")  
print("--------------------- Random Forest -----------------------")  
  
# Random Forest  
# Split data into training and test sets  
X = df\_merged.drop(['단속건수', '단속동', '단속건수\_범주'], axis=1)  
y = df\_merged['단속건수\_범주']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Create and fit the Random Forest classifier  
rf = RandomForestClassifier(n\_estimators=100, random\_state=42)  
rf.fit(X\_train, y\_train)  
  
# Predict test set labels  
y\_pred = rf.predict(X\_test)  
  
# Evaluate test-set accuracy  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"Random Forest Test set accuracy: {accuracy:.2f}")  
  
evaluate\_Kfold(X, y, 5, "Random Forest")  
evaluation\_classification(y\_test, y\_pred)  
classification\_report\_result = classification\_report(y\_test, y\_pred)  
print("Classification Report:")  
print(classification\_report\_result)  
  
  
print("--------------------- XGBoost -----------------------")  
  
# XGBoost  
# Split data into training and test sets  
X = df\_merged.drop(['단속건수', '단속동', '단속건수\_범주'], axis=1)  
y = df\_merged['단속건수\_범주'].map({'많음': 1, '적음': 0})  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Create and fit the XGBoost classifier  
xgb = XGBClassifier(random\_state=42)  
xgb.fit(X\_train, y\_train)  
  
# Predict test set labels  
y\_pred = xgb.predict(X\_test)  
  
# Evaluate test-set accuracy  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"XGBoost Test set accuracy: {accuracy:.2f}")  
  
evaluate\_Kfold(X, y, 5, "XGBoost")  
evaluation\_classification(y\_test, y\_pred)  
classification\_report\_result = classification\_report(y\_test, y\_pred)  
print("Classification Report:")  
print(classification\_report\_result)

**<Output>**

|  |
| --- |
| Print 성남시 교통단속 데이터 and 성남시 인구수 데이터 - 전처리 |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 흑백, 블랙이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print preprocessing dataset, merged dataset |
| 텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print 서울, 경기 상권 dataset and after preprocessing |
| 텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print MinMaxScaling |
| 텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print Prediction result |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print Ridge Regression Analysis - Evaluation |
| 텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print LinearRegression Analysis - Evaluation |
| 텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print Lasso Regression Analysis - Evaluation |
| 텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print Ridge Regression Analysis – Evaluation – Each column |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print category result |
| 텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print Decision Tree result |
| 텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |
|  |
| Print KNN, Random Forest, XG Boost result |
| 텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Print final\_decision\_tree.png |
| 텍스트, 스크린샷, 직사각형, 도표이(가) 표시된 사진  자동 생성된 설명 |

**<Object setting>**

Correlation analysis by analyzing illegal parking data and market data in Seoul and Seongnam.

It predicts that certain commercial districts will have a high rate of illegal parking, so it aims to find a specific market and analyze the reason to resolve the cause of illegal parking.Correlation analysis by analyzing illegal parking data and market data in Seoul and Seongnam.

It predicts that certain commercial districts will have a high rate of illegal parking, so it aims to find a specific market and analyze the reason to resolve the cause of illegal parking.

* Feedback

The initial purpose was to predict the number of future illegal parking in Seongnam-si with only Seongnam-si data, but when Seongnam-si data were collected and integrated based on this standard, there were only 14 records in the final data set that had been preprocessed.

So, the datasets are organized more neatly in units of dong, and the Seoul Metropolitan Government's datasets around Seongnam City are collected and used for prediction.

Through this process, predictions that were originally made only with 14 records can be predicted through 190 records. Of course, 190 records may not be enough data for data analysis, but efforts have been made to increase the number of records to a level that can be called analysis.

**<Using library>**

Pandas: A library for manipulating and analyzing data, useful for handling tabular data. Provides a data structure called DataFrame to facilitate data readability and processing.

numpy (numpy): library for numerical computation, specialized in multidimensional array and matrix operations. It is useful for mathematical calculations and arrangement processing.

Matplotlib.pyplot (Mattplotlib): A library for data visualization, generating graphs and providing various forms of visualization. It is mainly used to visually represent the distribution, relationship, trend, etc. of data.

seaborn: Matplotlib-based data visualization library for simpler, more sophisticated visualizations. It is mainly useful for dealing with statistical graphs, distribution visualizations, correlations, and more.

graphviz (graphbeads): Graph visualization tool, used to visually represent models such as decision trees.

sklearn.model\_selection: A module for selecting and evaluating models provided by the psychedelic run. Used primarily to segment data into training and test sets or to perform cross-validation.

sklearn.linear\_model (Sikitrun Linear Model): A module that contains models related to linear regression provided by the Sikitrun. You can implement linear regression, ridge regression, and Lasso regression.

sklearn.preprocessing: A module for data preprocessing that allows you to scale, encode, and transform data.

sklearn.tree (Sikitrun Decision Tree): A module that contains the decision tree algorithm provided by the Sikitrun. You can implement a decision tree model used for classification and regression.

sklearn.metrics: a module that calculates indicators for evaluating the performance of a model. It provides various metrics such as accuracy, confusion matrix, precision, reproduction rate, and F1 score.

sklearn.ensemble (Cykitrun Ensemble Model): a module containing an ensemble model provided by the cykitrun. Various ensemble models such as random forest, gradient boosting, and ada boost can be implemented.

xgboost.XGBCclassifier: XGBCclassifier, a classification model provided by the XGBoost library. XGBoost is a gradient boosting-based algorithm with excellent performance and scalability.

Matplotrip.font\_manager, rc (Mattplotlib font manager, rc): module and how to set up Korean fonts in Mattplotlib. Specify Korean fonts to help you use Korean text in your graph.

**<Tried algorithms>**

1. Linear Regression : Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship and calculates the best-fit line that minimizes the sum of squared differences between the observed and predicted values.

2. Ridge Regression : Ridge regression is a variant of linear regression that addresses multicollinearity (high correlation between independent variables) and helps prevent overfitting. It adds a penalty term to the sum of squared differences, which controls the magnitude of the coefficients. This penalty term, controlled by a regularization parameter (lambda), shrinks the coefficients towards zero, reducing their variance. This regularization helps stabilize the model and can improve its generalization performance by balancing bias and variance.

3. Lasso Regression : Lasso regression is a variation of linear regression that performs both variable selection and regularization. It adds a penalty term to the sum of squared differences, similar to ridge regression, but with a different penalty function. Lasso regression encourages sparsity in the coefficient estimates by shrinking some coefficients to exactly zero. This property makes lasso regression useful for feature selection and creating simpler models.

1.Decision Tree:

Intuitive interpretation of results: The decision tree represents decision rules for classification or regression problems in a tree structure, which provides an intuitive understanding of decision rules at each branch point. This helps you understand the patterns and influences of your data.

Determining the importance of a variable: The decision tree lets you evaluate the importance of a variable. Variables at the top of the tree are considered variables that have a greater impact on prediction. This allows you to identify important variables and increase the explanatory power of the model.

Nonlinear Relationship Modeling: The decision tree has strengths in modeling nonlinear relationships. It can express nonlinear patterns or interactions that are difficult to express well with other algorithms.

2.XGBoost (eXtreme Gradient Boosting):

Excellent predictive performance: XGBoost is based on the gradient boosting algorithm, and provides high predictive performance through ensemble learning. XGBoost typically shows better prediction than other algorithms.

Automatic Attribute Selection: XGBoost automatically evaluates the importance of a variable, allowing you to focus on important attributes. You can leverage the importance of variables to eliminate unnecessary variables or identify characteristics that contribute to model improvement.

Prevention of overfitting: XGBoost can apply Regulatoryization techniques to prevent overfitting. It prevents the model from overfitting to the learning data and improves generalization.

3.Random Forest:

Reliable predictive performance: Random Forest performs predictions by combining multiple decision trees, providing more reliable predictive performance than a single decision tree. This averages the prediction results of multiple trees, reducing overfitting and improving generalization performance.

Estimate the importance of a variable: Random forests can easily estimate the importance of a variable. Variable importance can be used to assess the importance of the variables used in the modeling and use them for variable selection, feature engineering, and model description.

Flexibility and diversity: Random forests can be applied to a wide variety of data and can handle both categorical and continuous variables. In addition, random forests are robust against outliers and handle well on large datasets.

4. KNN

A nonparametric machine learning algorithm used to solve classification and regression problems. KNN predicts based on neighbors at a given data point

KNN is a simple yet useful algorithm that is widely used in machine learning modeling to understand and predict the features and patterns of data. However, large datasets can result in higher computational costs, and higher dimensions of data can lead to higher-dimensional curses

**<Dataset Used>**

경기도 성남시\_주정차 위반 단속 위치 현황

<https://www.data.go.kr/data/15037104/fileData.do>

서울특별시 강동구\_동별 불법주정차 단속현황

<https://www.data.go.kr/data/15081659/fileData.do>

서울특별시 강서구\_불법주정차 단속 현황

<https://www.data.go.kr/data/15083768/fileData.do>

서울특별시 구로구\_주정차단속현황

<https://www.data.go.kr/data/15034492/fileData.do>

서울특별시 서초구\_주정차 단속 현황

<https://www.data.go.kr/data/15087185/fileData.do>

서울특별시 성북구\_불법주정차 동별 데이터

<https://www.data.go.kr/data/15113658/fileData.do>

서울특별시 송파구\_주정차단속건수정보

<https://www.data.go.kr/data/15048835/fileData.do>

서울특별시 영등포구\_주정차단속현황

<https://www.data.go.kr/data/15034483/fileData.do>

서울특별시 용산구\_불법주정차단속현황

<https://www.data.go.kr/data/15084175/fileData.do>

서울특별시 종로구\_불법주정차 통계

<https://www.data.go.kr/data/15100293/fileData.do>

서울특별시\_강남구\_불법주정차단속현황

<https://www.data.go.kr/data/15048827/fileData.do>

서울특별시\_서대문구\_주정차 단속 현황

<https://www.data.go.kr/data/15034465/fileData.do>

* After preprocessing : 단속동 (object) , 단속건수 (float)

-------------------------------------------------------------------------------------

소상공인시장진흥공단\_상가(상권)정보

<https://www.data.go.kr/data/15083033/fileData.do>

* After preprocessing : 행정동명 (object), 상권업종대분류명 (categorical data)

-------------------------------------------------------------------------------------

경기도 성남시\_인구및세대\_현황

<https://www.data.go.kr/data/15007386/fileData.do>

서울 인구수 데이터

<https://www.data.go.kr/data/15046938/fileData.do>

* After preprocessing : 행정동명 (object), 인구 (명) (int)

**-Description of the data column to use after preprocessing**

단속동: Datatype: String

단속건수: Datatype: Float

인구(명): Datatype: Float

상권업종대분류명\_관광/여가/오락

상권업종대분류명\_부동산

상권업종대분류명\_생활서비스

상권업종대분류명\_소매

상권업종대분류명\_숙박

상권업종대분류명\_스포츠

상권업종대분류명\_음식

상권업종대분류명\_학문/교육

-----------🡪 The above data are extracted from each data set.

단속건수, 단속동, 인구(명): Used to analyze the number and region of illegal parking crackdowns

상권업종대분류명\_etc: Used to analyze commercial districts around crackdown locations – (original data type is categorical Object)

The name of the commercial sector's large classification has eight categories, and it was changed to an int value through one-hot encoding.  
It integrates and analyzes the number of crackdowns and crackdowns included in the illegal parking crackdown data set and market data for the area. It is used to analyze specific markets and analyze which markets will have a high illegal parking rate in the future, expecting that the illegal parking rate will be high near certain markets.

텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

**<PPT Detail>**

Participated in Seongnam-si public data utilization and analysis contest in the objective setting part. In the process of finding a problem in Seongnam, we found that illegal parking was the biggest problem.

The previous work was to use Seongnam-si public datasets and associate the column values of each dataset so that it was possible to obtain illegal parking stop prediction values only for the desired year and region in Seongnam-si using regression.

However, since it was difficult to obtain meaningful predicted values with only the Seongnam-si public data set, the public data of Seoul, which has a similar location and characteristics to Seongnam-si, was additionally used to correlate the column values of the public data set of Seoul. The number of illegal parking in Seongnam was predicted. Seongnam-si and Seoul-si public data sets were obtained from DATA.Go.Kr

Output each data set obtained from Data Inspection to see if each column and its corresponding data type have the correct values. If not, change or remove the data values.

In Data Preprocessing, Extract only rows with ‘dong’ or ‘ro’ in the place name (the name of the column indicating the place is different for each dataset) and Match place names with words in the 'words\_to\_check' list to unify places by dong. Finally Merge data with the same place name.

Most of the data sets consisted of the number of enforcement cases and the population based on the enforcement dong especially 8 categories in the name of commercial sector classification. After creating a column with one-hot encoding, Each column has an int by summing on the basis of punctuated motion After that, scaling is performed using MinMaxScaler.

In data analysis, regression and classification were used, and in regression, the rate of change in population size and expected year were input from the user, and the result value for the desired number of illegal parking was displayed.

In classification, the mean value of the number of enforcement cases is obtained by associating the data set with the number of crackdown cases, the number of crackdown cases, the population name column, the commercial area classification name, and the number of crackdown cases. I got 193 row values and 12 columns. In addition, decision tree technique was used and uml diagram was also implemented.

In order to evaluate regression in data evaluation, it was evaluated using Linear, Ridge, and Lasso algorithms, and as a result of the evaluation, it was implemented to output MSE, MAE, and MAE values, respectively. As a result, the Ridge algorithm outputted the least error rate among the three algorithms. found the most suitable

In addition, to evaluate classification, it was evaluated using Decision Tree, KNN (using K Fold), Random Rorest, and XGBoost algorithms, and the evaluation results were implemented to output Test set accuracy, Average accuracy, and Accuracy values, respectively. As a result, the Random Forest algorithm This was the most accurate output, so the most suitable of the four algorithms was found.

As a result, using the public data set of Seongnam-si and Seoul-si, each column value was correlated and regression and classification were used to obtain the prediction value of illegal parking in Seongnam-si.

**<GitHub>**

<https://github.com/1109min/DataScience-TermProject-Team11>

**<Learning Experience>**

이승민 : I spent a lot of time finding the dataset and preprocessing it, and I realized that it's not so much the difficulty of preprocessing itself but the challenge of planning the overall flow. I learned that preprocessing takes a long time because it involves understanding the entire process and considering various algorithms that can be useful and determining which algorithm is best to use. When I saw the results of data analysis, I was fascinated, and it sparked my interest. Although it was challenging to do the end-to-end process of data analysis in a team project, I realized that each step is interconnected, and it took a long time to understand each other's code. It was also a valuable experience in terms of collaboration strategies.

정성수 : From the start, I learned how important it is to set a definite goal and find data to match it. Of course, our group set goals and found datasets for this purpose, but it was not to find exactly which data to predict, but to gather and predict datasets that looked good, so we had to think about whether the goals were shaken or how to use them again. Therefore, if I have a chance to do a similar project next time, I will solidify what data I will predict exactly from the beginning.

장현우 : Through this project, I realized the importance of which dataset to set. In addition, in the process of finding out the problems of Seongnam City by applying for the Seongnam City Public Data Utilization Contest and finding out why the problems occurred and what factors affected them, various algorithms learned in class were applied and other algorithms that were not learned in class were applied. I learned a lot in the process of finding and studying and comparing the accuracy with the algorithms I had learned before, and finding which algorithm was suitable. In addition, even after finding a suitable dataset, we found that many records can be found to obtain meaningful results only when certain columns are correlated well in the process of finding new records. In addition, while the project was in progress, the team members constantly communicated with each other and created a high level of unity.

최현준 : Through this project, I learned the importance of data. No matter how many datasets exist and how large they are, they were useless if proper preprocessing did not proceed. In fact, the data analysis we conducted used a lot of datasets, but not much data was actually left after the preprocessing process. I thought it was caused by our lack of skills and experience and not being able to properly process preprocessing and organize the right data, and I thought it took more time and effort than I thought to organize the data. It took a lot of time to apply the theoretically learned concept, and I think it is a part that needs to be studied more and more.

**<Teamwork>**

이승민 : data preprocessing, data analysis, data curation – feedback, final report (30%)

정성수 : data inspection, data preprocessing, data analysis, final report (30%)

장현우 : data evaluation, ppt, ppt – feedback (20%)

최현준 : data curation, data evaluation, presentation, final report (20%)