MỤC LỤC

NỘI DUNG

I) GIỚI THIỆU ĐỀ TÀI:

1) Phân công nhiệm vụ:

- Nhóm 11 và có 4 thành viên đang học ngành Khoa học Dữ liệu, lớp 20KDL1 tại trường Đại học Khoa Học Tự nhiên thành phố Hồ Chí Minh.

MSSV	Họ và tên	Công việc
20280109	Mai Chí Trung	Thuyết trình
20280084	Mai Chí Thanh	Nội dung
20280045	Nguyễn Quốc Huy	Nội dung
20280039	Đặng Ngọc Hưng	Nội dung

2) Mục đích chọn đề tài nghiên cứu:

- Theo UNICEF, thực trạng ô nhiễm môi trường hiện nay khá nóng và đang diễn ra ở khắp nơi trên thế giới, đặc biệt tại các nước đang phát triển, trong đó có Việt Nam. Bên cạnh ô nhiễm môi trường đất, nước do chất thải từ các công ty, xí nghiệp chưa qua xử lý đúng quy trình bị xả thẳng ra môi trường thì chất thải, khói bụi từ hoạt động của các nhà máy sản xuất, khí thải từ các phương tiện giao thông,... đã làm cho chất lượng không khí, đặc biệt tại các thành phố lớn, giảm sút và ô nhiễm đáng kể gây ra ô nhiễm không khí khá trầm trọng. Ô nhiễm không khí làm cho mọi người phải tiếp xúc với các hạt mịn trong không khí bị ô nhiễm. Các hạt mịn này thâm nhập sâu vào phổi và hệ thống tim mạch, gây ra các bệnh đột quỵ, bệnh tim, ung thư phổi, bệnh phổi tắc nghẽn mãn tính và các bệnh nhiễm trùng đường hô hấp, ảnh hưởng tới các nền kinh tế và chất lượng cuộc sống của con người.
- Ô nhiễm không khí đe dọa sức khỏe của người dân ở khắp mọi nơi trên thế giới. Năm 2018 ước tính cho thấy rằng 9/10 người dân phải hít thở không khí chứa hàm lượng các chất gây ô nhiễm cao. Ô nhiễm không khí cả ở bên ngoài và trong nhà gây ra khoảng 7 triệu ca tử vong hàng năm trên toàn cầu; chỉ tính riêng khu vực Tây Thái Bình Dương, khoảng 2,2 triệu người tử vong mỗi năm. Ở Việt Nam, khoảng 60.000 người chết mỗi năm có liên quan đến ô nhiễm không khí.
- Vì vậy nhóm quyết định thu thập dữ liệu về chất lượng không khí ở các thành phố lớn, cụ thể là thành phố Hồ Chí Minh, để có những cái nhìn về chất lượng không khí nơi đó cũng như tại các thành phố lớn trên thế giới.

II) Quá trình thu thập dữ liệu

1) Chuẩn bị dữ liệu:

- Dữ liệu được nhóm em thu thập từ : https://openweathermap.org thông qua do trang đã cung cấp sẵn API: http://api.openweathermap.org/data/2.5/air_pollution/history?lat=10.7546664&lon=106.415029&start=946726894&end=1610129052&appid=9f7c977884b479967cee75682a860b6b
- **OpenWeatherMap** là một dịch vụ trực tuyến, thuộc sở hữu của OpenWeather Ltd, cung cấp dữ liệu thời tiết toàn cầu thông qua API , bao gồm dữ liệu thời tiết hiện tại, dự báo , dự báo và dữ liệu thời tiết lịch sử cho bất kỳ vị trí địa lý nào. Công ty cung cấp dự báo lượng mưa siêu địa phương từng phút cho bất kỳ địa điểm nào. Mô hình học máy tích hợp được sử dụng để sử dụng các dịch vụ phát sóng khí tượng và dữ liệu từ các trạm thời tiết sân bay , trạm radar trên mặt đất , vệ tinh thời tiết , vệ tinh viễn thám, METAR và các trạm thời tiết tự động .
- Sự đa dạng của các API thời tiết do OpenWeatherMap cung cấp đã nhận thấy sự phổ biến đáng kể trong số các nhà phát triển phần mềm , dẫn đến việc ngày càng có nhiều kho lưu trữ trên GitHub . API hỗ trợ nhiều ngôn ngữ , đơn vị đo lường và định dạng dữ liệu tiêu chuẩn ngành như JSON và XML.

2) Làm việc với dữ liệu đã thu thập được:

- Import các thư viện cần thiết

```
%matplotlib inline
import matplotlib.pyplot as plt
import requests
import json
import pandas as pd
import numpy as np
import csv

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import make_pipeline
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import r2_score
```

- Sử dụng API từ openweathermap.org lưu vào biến data. Lấy dữ liệu chất lượng không khí tại thành phố Hồ Chí Minh từ năm 2018-2021. Nhận được 1074 bộ dữ liệu (~ 3 năm).

```
data=requests.get('http://api.openweathermap.org/data/2.5/air_pollution/history?lat=10.7546664&lon=106.415029&s
```

- API trả về dữ liệu dạng JSON

```
#Parse JSON
air_quality=json.loads(data.text)

#Test
air_quality['list'][0]
```

```
{'main': {'aqi': 3},
 'components': {'co': 727.65,
  'no': 3.07,
  'no2': 11.14,
  'o3': 10.1,
  'so2': 2.53,
  'pm2_5': 22.03,
  'pm10': 29.62,
  'nh3': 7.09},
 'dt': 1606266000}
```

- Dùng dict để biểu diễn dữ liệu và làm phẳng dữ liệu

```
#Simplize dicts
def make_flat_dict(p):
    res={};
    components=p['components']

    res['dt']=p['dt']
    res['co']=components['co']
    res['no']=components['no']
    res['no2']=components['no2']
    res['o3']=components['o3']
    res['so2']=components['so2']
    res['pm2_5']=components['pm2_5']
    res['pm10']=components['pm10']
    res['nh3']=components['nh3']
    res['aqi']=p['main']['aqi']

    return res
    pass

# Make list records
list_records=[]
for record in air_quality['list']:
    list_records.append(make_flat_dict(record))
```

- Xem kích thước bộ dữ liệu

```
len(list_records)
```

```
1074
```

- Lưu dữ liệu vào file air_quality.csv

```
#Write to csv
with open('air_quality.csv', 'w',newline='') as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=list_records[0].keys())
        writer.writeheader()
        for data in list_records:
            writer.writerow(data)
```

III) Khám phá dữ liệu:

1) Khám phá dữ liệu thu được:

- Đọc dữ liệu từ file đã thu thập trước đó

```
# Đọc dữ liệu từ file đã thu thập trước đó
data_df = pd.read_csv('air_quality.csv')
```

- Tổng quan dữ liệu thu được

```
data_df.head(10)
```

	dt	co	no	no2	o3	so2	pm2_5	pm10	nh3	aqi
0	1606266000	727.65	3.07	11.14	10.10	2.53	22.03	29.62	7.09	3
1	1606269600	761.03	5.70	13.54	15.38	5.96	21.35	31.40	9.50	3
2	1606273200	934.60	8.27	21.25	25.03	12.52	26.29	39.77	12.79	4
3	1606276800	594.14	3.80	15.42	76.53	12.99	20.08	28.83	7.28	3
4	1606280400	560.76	2.12	14.40	135.90	18.60	34.19	43.10	6.65	4
5	1606284000	654.22	2.01	19.19	191.69	35.76	63.55	73.70	7.28	5
6	1606287600	654.22	0.77	23.99	197.41	44.35	70.69	80.74	7.16	5
7	1606291200	627.52	0.69	29.82	165.94	42.44	55.14	65.30	7.28	5
8	1606294800	600.82	0.59	33.59	137.33	37.19	41.74	52.30	7.79	4
9	1606298400	667.57	0.48	38.04	103.00	26.46	34.72	46.30	13.30	4

- Xem dữ liệu có bao nhiêu dòng bao nhiêu cột

```
data_df.shape
```

```
(1074, 10)
```

- Xem dữ liệu có bị lặp không . Nếu các dòng bị lặp thì chỉ giữ lại dòng đầu tiên vì các dòng trùng không có tác dụng trong việc huấn luyện mô hình.

```
[40]: data_df.index.duplicated().sum()
```

```
[40]: 0
```

- Kết quả trả về giá trị 0. Vậy nên ta có đủ lượng dữ liệu cần và các dòng dữ liệu cũng không bị trùng lặp.

2) Tổng quan về dữ liệu:

Quan sát sơ bộ dữ liệu ta thấy mỗi dòng chứa số liệu của các loại chất khác nhau.

Ta có chỉ số chất lượng không khí là cột ***aqi*** . Các giá trị có thể nhận: 1, 2, 3, 4, 5. Trong đó:

- 1 = Tốt
- 2 = Khá
- 3 = Trung bình
- 4 = Kém
- 5 = Rất Kém

Bên cạnh chỉ số Air Quality Index (AQI), API cũng trả về hàm lượng các khí ô nhiễm như:

- CO: Carbon Monoxide
  - Phát sinh khi đốt C trong điều kiện thiếu O2, thường bắt nguồn từ việc đốt nhiên liệu hóa thạch, các phương tiện giao thông.
  - Làm giảm lượng oxy có thể được vận chuyển trong máu đến các cơ quan quan trọng như tim và não, có thể gây chóng mặt, nhầm lẫn, bất tỉnh và tử vong.
- NO,NO2: Nitrogen Monoxide,Nitrogen Dioxide

- Chủ yếu bay vào không khí từ quá trình đốt cháy nhiên liệu. NO2 hình thành từ khí thải của ô tô, xe tải và xe buýt, nhà máy điện và thiết bị ngoài đường.
- Gây kích ứng đường thở trong hệ hô hấp của con người,trong thời gian ngắn có thể làm nặng thêm các bệnh về đường hô hấp, đặc biệt là hen suyễn, dẫn đến các triệu chứng về đường hô hấp (như ho, khò khè hoặc khó thở).

- O3: Ozone

  - Nó được hình thành các phản ứng hóa học với sự có mặt của các chất ô nhiễm tiền chất như NOx và các hợp chất hữu cơ dễ bay hơi VOC.
  - Kích ứng và gây viêm mắt, mũi, họng và đường hô hấp dưới: ho, đau và cào họng hoặc cảm giác khó chịu ở ngực,giảm chức năng phổi: không thể thở sâu hoặc mạnh như bình thường, có thể tiếp tục làm hỏng phổi khi các triệu chứng đã biến mất.

- SO2: Sulphur Dioxide

  - Nhà máy điện và nhà máy lọc dầu, chất tẩy rửa khô, trạm dịch vụ xăng dầu và đốt gỗ dân dụng, phương tiện trên đường.
  - Gây ảnh hưởng xấu đến chức năng hô hấp, đặc biệt đối với những người mắc bệnh hen suyễn.

- NH3: Ammonia

- PM2.5 và PM10: bụi mịn có kích thước nhỏ hơn 2.5 và 10 μm

  - Kết quả của các phản ứng phức tạp của các hóa chất như sulfur dioxide và nitơ oxit, là những chất gây ô nhiễm phát ra từ các nhà máy nhiệt điện, công nghiệp và phương tiện giao thông.
  - Gây ra các vấn đề sức khỏe nghiêm trọng.

Đến đây, ta thấy dữ liệu đã ổn và sẵn sàng cho việc đưa ra câu hỏi về bộ dữ liệu.

IV) Tách các tập dữ liệu:

1) Kiểm tra cột dữ liệu output:

- Chúng ta thực hiện các thao tác kiểm tra như: cột output có kiểu dữ liệu gì, có giá trị thiếu không, tỉ lệ các lớp trong cột output.

- Sau khi thực hiện kiểm tra, chúng ta thấy rằng cột output đang ở dạng số và không bị thiếu dữ liệu. Tỉ lệ các lớp cũng không vấn đề gì.

2) Tiền xử lý (tách các tập):

Bây giờ ta sẽ thực hiện bước tiền xử lý là tách tập validation và tập kiểm tra ra. Ở đây, cột output của dữ liệu chính là cột 'aqi' (chất lượng không khí).

- Tách X và y

[50]:
```
y_sr = data_df["aqi"]
X_df = data_df.drop("aqi", axis=1)
```

- Tách dữ liệu theo tỉ lệ: 80% cho tập train, 10% cho tập validation, 10% cho tập test.

[54]:
```
# Tách tập (train, validation) và tập test
new_X_df, test_X_df, new_y_sr, test_y_sr = train_test_split(X_df, y_sr, test_size = 0.1,
                                                            stratify=y_sr, random_state=50)

# Tách tập train và tập validation
train_X_df, val_X_df, train_y_sr, val_y_sr = train_test_split(new_X_df, new_y_sr, test_size=(1/9),
                                                              stratify=new_y_sr, random_state=50)
```

[44]:
```
train_X_df.shape
```

[44]: (858, 9)

[45]:
```
train_y_sr.shape
```

[45]: (858,)

[46]:
```

val_X_df.shape
```

[46]: (108, 9)

[47]:
```

val_y_sr.shape
val_y_sr.shape
```

[47]: (108,)

[48]:
```
test_X_df.shape
```

[48]: (108, 9)

[49]:
```
test_y_sr.shape
```

[49]: (108,)

3) Tập huấn luyện dữ liệu:

- Kiểm tra dữ liệu thu được đang ở dạng gì?

```
[55]: train_X_df.dtypes
```

```
[55]: dt         int64
      co       float64
      no       float64
      no2      float64
      o3       float64
      so2      float64
      pm2_5    float64
      pm10     float64
      nh3      float64
      dtype: object
```

Như vậy, dữ liệu thu thập đều ở dạng mong muốn mà không phải xử lý gì thêm. Ở đây, tất cả các cột đều ở dạng số.

- Dữ liêu được phân bố ra sao

```
[57]: def missing_ratio(df):
          return (df.isna().mean() * 100).round(1)
      def lower_quartile(df):
          return df.quantile(0.25).round(1)
      def median(df):
          return df.quantile(0.5).round(1)
      def upper_quartile(df):
          return df.quantile(0.75).round(1)
      train_X_df.agg([missing_ratio, 'min', lower_quartile, median, upper_quartile, 'max'])
```

[57]:

	dt	co	no	no2	o3	so2	pm2_5	pm10	nh3
missing_ratio	0.000000e+00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
min	1.606266e+09	283.72	0.00	1.21	0.00	0.95	4.55	5.45	1.54
lower_quartile	1.607239e+09	480.60	0.00	6.70	13.40	2.50	14.90	18.50	4.80
median	1.608208e+09	680.90	0.20	14.70	40.80	4.60	30.30	37.60	7.60
upper_quartile	1.609188e+09	1158.20	1.30	28.80	65.80	13.10	59.70	71.30	11.80
max	1.610129e+09	4592.90	89.41	131.61	246.05	82.02	240.65	275.21	94.24

- Như vậy là dữ liệu không có giá trị thiếu. Có thể thấy nguồn dữ liệu là rất tin cậy và đầy đủ.

4) Tiền xử lý tập huấn luyện :

Với tập dữ liệu thu được, ta nhận thấy cột dt (Date and Time) ghi nhận số giây từ 1/1/1970 đến thời điểm thu thập dữ liệu là không cần thiết với việc đánh giá chất lượng không khí nên ta sẽ thực hiện xóa cột này đi. Việc này sẽ thực hiện trong hàm ColDropper.

[58]:
```
class ColDropper(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass
    def fit(self, X_df, y=None):
        return self
    def transform(self, X_df, y=None):
        df = X_df.drop('dt', axis=1)
        return df
```

[59]:
```
col_dropper = ColDropper()
col_dropper.fit(train_X_df)
```

[59]:
```
ColDropper()
```

[60]:
```
#Test
fewer_cols_train_X_df = col_dropper.transform(train_X_df)
fewer_cols_train_X_df.head(5)
```

[60]:

	co	no	no2	o3	so2	pm2_5	pm10	nh3
974	1508.71	1.62	61.69	2.12	21.46	67.04	90.39	27.87
854	453.95	0.00	12.17	55.79	10.13	14.63	18.49	4.31
211	347.14	0.00	4.88	61.51	2.33	21.21	29.40	6.14
1054	720.98	0.18	16.97	9.48	2.33	24.32	31.66	13.30
980	794.41	0.05	16.79	15.20	2.27	26.06	29.56	7.73

V) Mô hình hoá:

1) Thử nghiệm với mô hình MLP Classifier:

Tạo pipeline với các với các siêu tham số MLPClassifier gồm hidden_layer_sizes=(20), activation='tanh', solver='lbfgs', random_state=0, max_iter=2500) . Điền các giá trị thiếu bằng giá trị trung bình.

[67]:
```
pipeline = make_pipeline(ColDropper(),
                         SimpleImputer(missing_values=np.nan, strategy='mean'),
                         StandardScaler(),
                         MLPClassifier(hidden_layer_sizes=(20), activation='relu', solver='lbfgs', random_state=0,
                                       max_iter=2500))
```

```
# Thử nghiệm với các giá trị khác nhau của các siêu tham số
# và chọn ra các giá trị tốt nhất
train_errs = []
val_errs = []
alphas = [0.01, 0.1, 1, 10, 100, 1000]
best_val_err = float('inf'); best_alpha = None
for alpha in alphas:
    pipeline.set_params(mlpclassifier__alpha=alpha)
    pipeline.fit(train_X_df, train_y_sr)
    train_errs.append((1 - pipeline.score(train_X_df, train_y_sr))*100)
    val_errs.append((1 - pipeline.score(val_X_df, val_y_sr))*100)
```

```
    val_errs.append((1 - pipeline.score(val_X_df, val_y_sr))*100)
    if val_errs[-1] <= best_val_err:
        best_val_err = val_errs[-1]
        best_alpha = alpha
'Finish!'
```

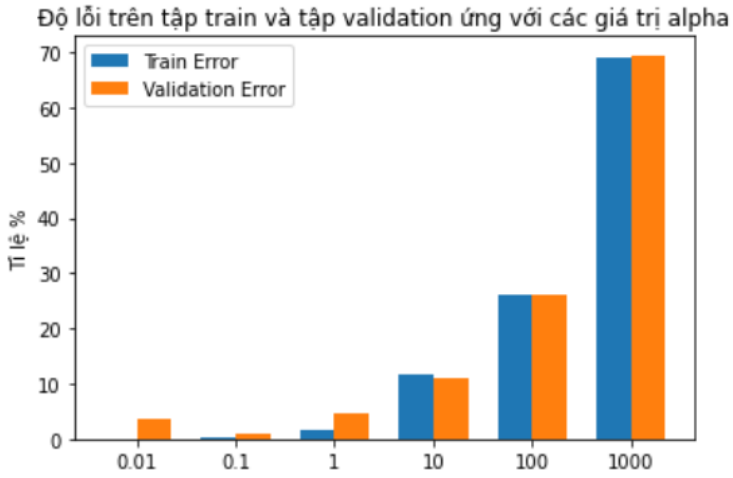
```
'Finish!'
```

```
ser1 = pd.Series(train_errs)
ser2 = pd.Series(val_errs)

ind = np.arange(len(ser1))
width = 0.35
fig, ax = plt.subplots()
ax.bar(ind - width/2, ser1, width, label='Train Error')
ax.bar(ind + width/2, ser2, width, label='Validation Error')

ax.set_xlabel('Alphas')
ax.set_ylabel('Tỉ lệ %')
ax.set_title('Độ lỗi trên tập train và tập validation ứng với các giá trị alpha')
ax.set_xticks([0, 1, 2, 3, 4, 5])
ax.set_xticklabels(alphas)
ax.legend()

plt.show()
```




```
print("Best validation error: " + str(best_val_err.round(2)) + '%')
print("Best alpha: " + str(best_alpha))
```

```
Best validation error: 0.93%
Best alpha: 0.1
```

- Tạo full pipeline với best alpha vừa tìm được, huấn luyện trên tập train + tập validation

```
full_pipeline = pipeline.set_params(mlpclassifier__alpha=best_alpha)
full_pipeline.fit(new_X_df, new_y_sr)
```

```
Pipeline(steps=[('coldropper', ColDropper()),
                ('simpleimputer', SimpleImputer()),
                ('standardscaler', StandardScaler()),
                ('mlpclassifier',
                 MLPClassifier(alpha=1, hidden_layer_sizes=20, max_iter=2500,
                               random_state=0, solver='lbfgs'))])
```

- Độ lỗi trên tập huấn luyện

```
(1 - full_pipeline.score(new_X_df, new_y_sr))*100
```

```
1.552795031055898
```

- Độ lỗi trên tập test

```
(1 - full_pipeline.score(test_X_df, test_y_sr))*100
```

```
7.4074074074074066
```

- Chỉ số thống kê của mô hình:

MLP classifier

```
print(classification_report(test_y_sr, y_preds))
```

	precision	recall	f1-score	support
1	0.09	1.00	0.17	10
2	0.00	0.00	0.00	27
3	0.00	0.00	0.00	8
4	0.00	0.00	0.00	30
5	0.00	0.00	0.00	33
accuracy			0.09	108
macro avg	0.02	0.20	0.03	108
weighted avg	0.01	0.09	0.02	108

- Nhận xét:
  Với mô hình huấn luyện này, độ lỗi thu được trên tập test là ổn và có thể chấp nhận được. Suy ra mô hình phân lớp khá tốt khi chọn siêu tham số alpha một cách hợp lí. Nhưng độ chính là f1-score của mô hình khá thấp nên đây chưa phải là mô hình thực sự tốt lắm.

2) Thử nghiệm với mô hình K-Neighbors Classifier:

- Tạo pipeline, điền các giá trị thiếu bằng giá trị trung bình.

[75]:
```
pipeline2 = make_pipeline(ColDropper(),
                          SimpleImputer(missing_values=np.nan, strategy='mean'),
                          StandardScaler(),
                          KNeighborsClassifier())
```

[76]:
```
# Thử nghiệm với các giá trị và chọn ra các số neighbors tốt nhất
train_errs = []
val_errs = []
neighbors = [1, 3, 5, 10, 20, 30, 50]
best_val_err = float('inf'); best_neighbor = None
for neighbor in neighbors:
    pipeline2.set_params(kneighborsclassifier__n_neighbors = neighbor)
    pipeline2.fit(train_X_df, train_y_sr)
    train_errs.append((1 - pipeline2.score(train_X_df, train_y_sr))*100)
    val_errs.append((1 - pipeline2.score(val_X_df, val_y_sr))*100)
    if val_errs[-1] <= best_val_err:
        best_val_err = val_errs[-1]
        best_neighbor = neighbor
'Finish!'
```

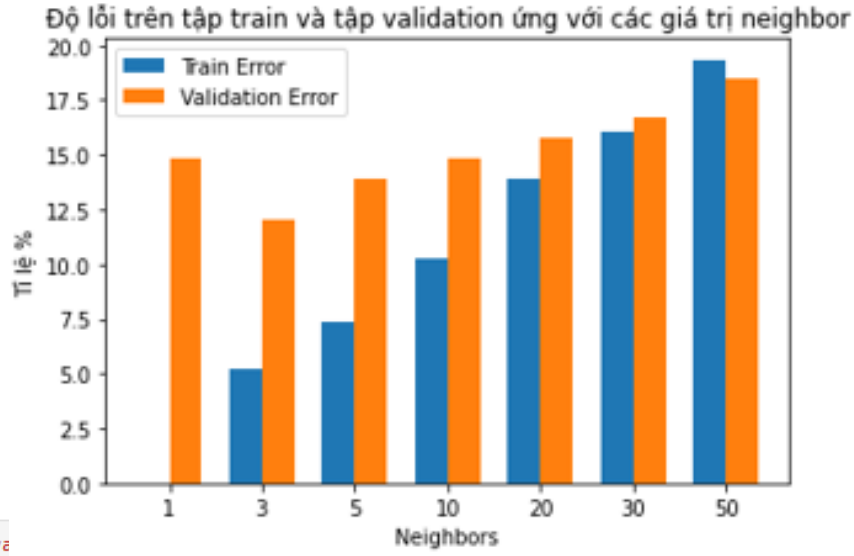
[76]: 'Finish!'

[77]:
```
ser1 = pd.Series(train_errs)
ser2 = pd.Series(val_errs)

ind = np.arange(len(ser1))
width = 0.35
fig, ax = plt.subplots()
ax.bar(ind - width/2, ser1, width, label='Train Error')
ax.bar(ind + width/2, ser2, width, label='Validation Error')

ax.set_xlabel('Neighbors')
ax.set_ylabel('Tỉ lệ %')
ax.set_title('Độ lỗi trên tập train và tập validation ứng với các giá trị neighbor')
ax.set_xticks([0, 1, 2, 3, 4, 5, 6])
ax.set_xticklabels(labels=neighbors)
ax.legend()

plt.show()
```



[78]:
```
print("Best va
print("Best neighbors: " + str(best_neighbor))
```

```
Best validation error: 12.04%
Best neighbors: 3
```

- Tạo full pipeline với best alpha vừa tìm được, huấn luyện trên tập train + tập validation.

```
[79]: full_pipeline2 = pipeline2.set_params(kneighborsclassifier__n_neighbors=best_neighbor)
      full_pipeline2.fit(new_X_df, new_y_sr)
```

```
[79]: Pipeline(steps=[('coldropper', ColDropper()),
                      ('simpleimputer', SimpleImputer()),
                      ('standardscaler', StandardScaler()),
                      ('kneighborsclassifier', KNeighborsClassifier(n_neighbors=3))])
```

- Độ lỗi trên tập huấn luyện

```
[87]: (1 -pipeline3.score(new_X_df, new_y_sr)) * 100
```

```
[87]: 0.0
```

- Độ lỗi trên tập test

```
[43]: (1- pipeline2.score(test_X_df, test_y_sr)) * 100
```

```
:[43]: 19.444444444444443
```

- Chỉ số thống kê mô hình:

KNeighborsClassifier

```
print(classification_report(test_y_sr, y_pred))
```

	precision	recall	f1-score	support
1	1.00	0.70	0.82	10
2	0.69	0.89	0.77	27
3	1.00	0.12	0.22	8
4	0.81	0.87	0.84	30
5	0.97	0.97	0.97	33
accuracy			0.83	108
macro avg	0.89	0.71	0.73	108
weighted avg	0.86	0.83	0.82	108

- Nhận xét**:**
  Như vậy độ lỗi trên tập test lớn hơn nhiều so với độ lỗi trên tập huấn luyện. Nhưng mô hình lại thực hiện khá tốt với 2 chỉ số precision và recall khá cao.

3) Thử nghiệm với mô hình Decision Tree Classifier:

- Tạo pipeline, điền các giá trị thiếu bằng giá trị trung bình.

[82]:
```
pipeline3 = make_pipeline(ColDropper(),
                          SimpleImputer(missing_values=np.nan, strategy='mean'),
                          StandardScaler(),
                          DecisionTreeClassifier())
```

[83]:
```
# Fit dữ liệu vào mô hình.
pipeline3.fit(new_X_df, new_y_sr)
```

[83]:
```
Pipeline(steps=[('coldropper', ColDropper()),
                ('simpleimputer', SimpleImputer()),
                ('standardscaler', StandardScaler()),
                ('decisiontreeclassifier', DecisionTreeClassifier())])
```

- Độ lỗi trên tập huấn luyện và tập validation:

[80]:
```
(1- pipeline2.score(new_X_df, new_y_sr)) * 100
```

[80]:
```
4.865424430641818
```

- Độ lỗi trên tập test:

[47]:
```
(1 - pipeline3.score(test_X_df, test_y_sr)) * 100
```

[47]:
```
0.92592592592593
```

- Chỉ số thống kê mô hình:

DecisionTreeClassifier

```
from sklearn.metrics import classification_report
print(classification_report(test_y_sr, y_preds))
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        10
           2       0.25      0.26      0.25        27
           3       0.00      0.00      0.00         8
           4       0.37      0.37      0.37        30
           5       0.36      0.36      0.36        33

    accuracy                           0.28       108
   macro avg       0.20      0.20      0.20       108
weighted avg       0.28      0.28      0.28       108
```

- Nhận xét: Vậy, trong trường hợp bài toán này thì Decision Tree thực hiện phân lớp với độ chính xác cao hơn nhiều trên tập test so với thuật toán MLP. Nhưng chỉ số f1-score và độ chính xác lại khá thấp.

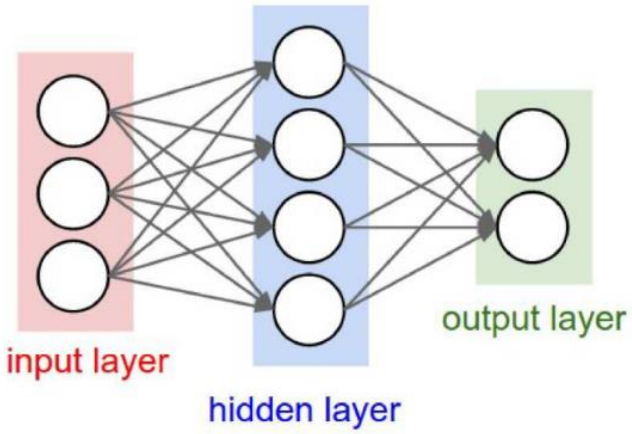
4) Kết luận:

- Các thuật toán nhóm đã chọn fit khá tốt với tập dữ liệu đã chọn. Tuy nhiên độ lỗi trên tập test của thuật toán K-Neighbors Classifier là khá cao (gần 20%) nhưng độ chính xác lại cao hơn so với 2 thuật toán MLPClassifier hay Decision Tree Classifier dựa trên chỉ số f1-score nên ta xem xét sử dụng K-Neighbors Classifier để đánh giá khi có tập test mới để có được dự đoán kết quả chất lượng không khí tin cậy nhất.

VI) Những thuật toán đã được áp dụng:

1) Mô hình MLP Classifier:

- Mạng Nơ-ron nhân tạo hay ngắn gọn là ANN ngày nay được sử dụng rộng rãi trong nhiều ứng dụng và phân loại là một trong số đó và cũng có nhiều thư viện và khuôn khổ dành riêng cho việc xây dựng Mạng Nơ-ron một cách dễ dàng.
- Mạng Neural là sự kết hợp của của các tầng perceptron hay còn được gọi là perceptron đa tầng (*multilayer perceptron*) như hình vẽ bên dưới



- MLPClassifier là viết tắt của Multi-layer Perceptron Classifier, trong tên gọi này chính nó kết nối với Mạng thần kinh. Không giống như các thuật toán phân loại khác như Hỗ trợ Vectơ hoặc Bộ phân loại Naive Bayes, MLPClassifier dựa vào Mạng nơ-ron bên dưới để thực hiện nhiệm vụ phân loại.
- Tuy nhiên, một điểm tương đồng với các thuật toán phân loại khác của Scikit-Learn là việc triển khai MLPClassifier không tốn nhiều công sức hơn việc triển khai Support Vectors hoặc Naive Bayes hoặc bất kỳ bộ phân loại nào khác từ Scikit-Learn.

- Hoạt động của mạng MLP như sau: tại tầng đầu vào các neural nhận tín hiệu vào xử lý (tính tổng trọng số, gửi tới hàm truyền) rồi cho ra kết quả (là kết quả của hàm truyền); kết quả này sẽ được truyền tới các neural thuộc tầng ẩn thứ nhất; các nơron tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến tầng ẩn thứ 2. Quá trình tiếp tục cho đến khi các neural thuộc tầng ra cho kết quả.

2) Mô hình K-Neighbors Classifier:

- K trong tên của bộ phân loại này đại diện cho k lân cận gần nhất, trong đó k là một giá trị nguyên do người dùng chỉ định. Do đó, như tên cho thấy, bộ phân loại này thực hiện việc học tập dựa trên k láng giềng gần nhất. Sự lựa chọn giá trị của k phụ thuộc vào dữ liệu.

- Trong thống kê, giải thuật K-Neighbors hay còn gọi là thuật toán k hàng xóm gần nhất, viết tắt từ tiếng Anh *k*-NN) là một phương pháp thống kê phi tham số (nonparametric statistics) được đề xuất bởi Thomas M. Cover để sử dụng cho phân loại bằng thống kê và phân tích hồi quy. Cụm từ *hàng xóm* có thể hiểu là *láng giềng* hoặc *lân cận*.

- Phân loại dựa trên lân cận là một loại *học tập dựa trên cá thể* hoặc học tập *không tổng quát hóa* : nó không cố gắng xây dựng một mô hình nội bộ chung, mà chỉ lưu trữ các phiên bản của dữ liệu đào tạo. Việc phân loại được tính toán từ đa số phiếu đơn giản của các lân cận gần nhất của mỗi điểm: một điểm truy vấn được chỉ định lớp dữ liệu có nhiều đại diện nhất trong các lân cận gần nhất của điểm.

- Scikit-learning triển khai hai bộ phân loại láng giềng gần nhất khác nhau: KNeighborsClassifier triển khai học tập dựa trên k hàng xóm gần nhất của mỗi điểm truy vấn, nơi k là một giá trị số nguyên do người dùng chỉ định. RadiusNeighborsClassifier triển khai học tập dựa trên số lượng hàng xóm trong bán kính cố định r của mỗi điểm đào tạo, ở đâu r là một giá trị dấu phẩy động do người dùng chỉ định.

- Phân loại láng giềng gần nhất cơ bản sử dụng trọng số đồng nhất: nghĩa là, giá trị được gán cho một điểm truy vấn được tính toán từ đa số phiếu bầu đơn giản của các láng giềng gần nhất. Trong một số trường hợp, tốt hơn là nên cân nhắc những người hàng xóm để những người hàng xóm gần hơn đóng góp nhiều hơn cho sự phù hợp. Điều này có thể được thực hiện thông qua weights từ khóa. Giá trị mặc định , gán trọng số đồng nhất cho mỗi hàng xóm. ấn định trọng số tỷ lệ với nghịch đảo của khoảng cách từ điểm truy vấn. Ngoài ra, một

chức năng do người dùng xác định về khoảng cách có thể được cung cấp để tính toán trọng số. weights = 'uniform'weights = 'distance'

3) Mô hình Decision Tree Classifier:

- Decision Trees (DTs) là một phương pháp học tập có giám sát phi tham số được sử dụng để classification và regression. Mục đích là tạo ra một mô hình dự đoán giá trị của một biến mục tiêu bằng cách tìm hiểu các quy tắc quyết định đơn giản được suy ra từ các tính năng dữ liệu. Một cây có thể được coi là một phép gần đúng không đổi từng mảnh.
- DecisionTreeClassifier là một lớp có khả năng thực hiện phân loại nhiều lớp trên một tập dữ liệu.
- Giống như với các bộ phân loại khác, DecisionTreeClassifierlấy hai mảng đầu vào: một mảng X, thưa thớt hoặc dày đặc, có hình dạng chứa các mẫu huấn luyện và một mảng Y chứa các giá trị nguyên, hình dạng , giữ các nhãn lớp cho các mẫu huấn luyện:(n_samples, n_features)(n_samples,). Cây quyết định cũng có thể được áp dụng cho các bài toán hồi quy, sử dụng DecisionTreeRegressor lớp.
- Khi không có mối tương quan giữa các đầu ra, một cách rất đơn giản để giải quyết loại vấn đề này là xây dựng n mô hình độc lập, tức là một mô hình cho mỗi đầu ra, và sau đó sử dụng các mô hình đó để dự đoán độc lập từng đầu ra trong số n đầu ra. Tuy nhiên, vì có khả năng các giá trị đầu ra liên quan đến cùng một đầu vào tương quan với nhau, nên một cách thường tốt hơn là xây dựng một mô hình duy nhất có khả năng dự đoán đồng thời tất cả n đầu ra. Đầu tiên, nó yêu cầu thời gian đào tạo thấp hơn vì chỉ có một công cụ ước tính duy nhất được xây dựng. Thứ hai, độ chính xác tổng quát của công cụ ước lượng kết quả thường có thể được tăng lên.

VII) Lời kết:

- Bài báo cáo này nhóm chúng em/mình dựa trên trên tinh thần tiếp cận một vấn đề thực tế và cũng là một phần ứng dụng phổ biến của Machine Learning (máy học). Việc lọc ra các ngày không khí bị ô nhiễm ở các mức độ từ thấp đến cao trong thực tế các thuật toán xử dụng rất phức tạp, bộ dữ liệu khổng lồ, chúng khác xa với những gì mà chúng ta đang thực hiện trong bài báo cáo này. Trong bài báo cáo này chỉ đơn giản là thực hiện cách lấy dữ liệu, xử lý tập dữ liệu được nạp vào, thử nghiệm trên một số mô hình và chọn ra mô hình dự đoán kết quả chất lượng không khí đáng tin cậy nhất. Nhưng trong thực tế,

những người xử lý dữ liệu họ có dùng những cách này không hay họ dùng mô hình khác tối ưu hơn, hiệu quả hơn. Chúng ta thấy rõ rằng việc xử lý dữ liệu trong thực tế quả thông qua rất nhiều quy trình phức tạp. Vì vậy bài báo cáo này mang tính chất tham khảo, một phần nào đó giúp chúng ta hình dung được một mô hình nhỏ về xử lý dữ liệu. Nội dung được các thành viên nhóm tham khao, sau đó tổng hợp và chắc lọc. Nếu có gì sai sót hoặc thắc mắc mong thầy và các bạn góp ý để giúp chúng em/mình hiểu rõ hơn về môn học và giúp cho bài báo cáo này tốt hơn. Nhóm chúng em/mình xin cảm ơn thầy và các bạn đã quan tâm và tham khảo bài báo cáo cuối kỳ Nhập môn Khoa học Dữ Liệu.

TÀI LIỆU THAM KHẢO

- https://scikit-learn.org (Tham khảo các thuật toán phân lớp có trong đồ án).
- https://openweathermap.org/api (Tham khảo về ý nghĩa của các cột dữ liệu thu được)
- https://www.tutorialspoint.com/scikit_learn/scikit_learn_kneighbors_classifier.htm?key=MLP+Classifier
- https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_k_h%C3%A0ng_x%C3%B3m_g%E1%BA%A7n_nh%E1%BA%A5t