Assignment 5 Notebook

Full name : Yuhan Liu

Wisc Id : 9069436013

Test Setup details :

Jupyter verison : 4.3.0

In [2]:
```
import pandas as pd
# numpy has a lots of useful math related modules
import numpy as np
# Helpful function to display intermittent result
from IPython.display import display
```

Step 0 : Take a peek at the dataset

In [3]:
```
# Load the data and display only first five rows
# NOTE replace the following value with the actual path to the csv fil
e
data_file = "AQI.csv" # e.g ~/cs564/p5/AQI.csv"
display(pd.read_csv(data_file, nrows=5).head())
```

	State Code	County Code	Latitude	Longitude	Date Local	AQI	Address	State Name	County Name
0	55	59	42.504722	-87.8093	1997-04-28	82	CHIWAUKEE PRAIRIE, 11838 FIRST COURT	Wisconsin	Kenosha
1	55	59	42.504722	-87.8093	1997-05-10	33	CHIWAUKEE PRAIRIE, 11838 FIRST COURT	Wisconsin	Kenosha
2	55	59	42.504722	-87.8093	1997-05-16	25	CHIWAUKEE PRAIRIE, 11838 FIRST COURT	Wisconsin	Kenosha
3	55	59	42.504722	-87.8093	1997-05-22	25	CHIWAUKEE PRAIRIE, 11838 FIRST COURT	Wisconsin	Kenosha
4	55	59	42.504722	-87.8093	1997-05-28	46	CHIWAUKEE PRAIRIE, 11838 FIRST COURT	Wisconsin	Kenosha

Step 1 : Load the dataset

In [4]:
```
from sqlalchemy import create_engine
import sqlite3
# create a database where we'll load the dataset from the csv file
db_conn = create_engine('sqlite:///AirQualityIndex.db')
```

In [5]:
```
import string
# load in a batch of 5000 tuples. Modify this value to your needs
chunks = 5000
for data in pd.read_csv(data_file, chunksize=chunks,
 iterator=True, encoding='utf-8'):

 data = data.rename(columns={col: col.replace('-', ' ') for col in dat
a.columns})
 data = data.rename(columns={col: col.strip() for col in data.columns}
)
 data = data.rename(columns={col: string.capwords(col) for col in data
.columns})
 data = data.rename(columns={col: col.replace(' ', '') for col in data
.columns})

 data.to_sql('data', db_conn, if_exists='append')
```

Step 2 : Data Exploration

Find the average air quality index for each city recorded during the year 1997

In [6]:
```
avg_city_latitude = pd.read_sql_query("""SELECT AVG(Latitude) as avg_l
atitude,
                             CityName as city FROM data
                             WHERE "DateLocal" LIKE "1997-%"
                             GROUP BY city
                             ORDER BY avg_latitude DESC""", db_conn)

# Display the result (Note - the result 'avg_air_quality' is an instan
ce of 'pandas DataFrame')
display(avg_city_latitude)
```

	avg_latitude	city
0	43.020075	Waukesha
1	43.016667	Milwaukee
2	42.504722	Pleasant Prairie

In [7]:
```
avg_city_longitude = pd.read_sql_query("""SELECT AVG(Longitude) as avg
_longitude,
                            CityName as city FROM data
                            WHERE "DateLocal" LIKE "1997-%"
                            GROUP BY city
                            ORDER BY avg_longitude DESC""", db_conn)

# Display the result (Note - the result 'avg_air_quality' is an instan
ce of 'pandas DataFrame')
display(avg_city_longitude)
```

	avg_longitude	city
0	-87.809300	Pleasant Prairie
1	-87.933333	Milwaukee
2	-88.215070	Waukesha

In [8]:
```
max_air_quality = pd.read_sql_query("""SELECT MAX(AQI) as max_aqi,
                            CityName as city, DateLocal as date FROM
data
                            WHERE "DateLocal" LIKE "1997-%"
                            GROUP BY city
                            ORDER BY max_aqi DESC""", db_conn)

# Display the result (Note - the result 'avg_air_quality' is an instan
ce of 'pandas DataFrame')
display(max_air_quality)
```

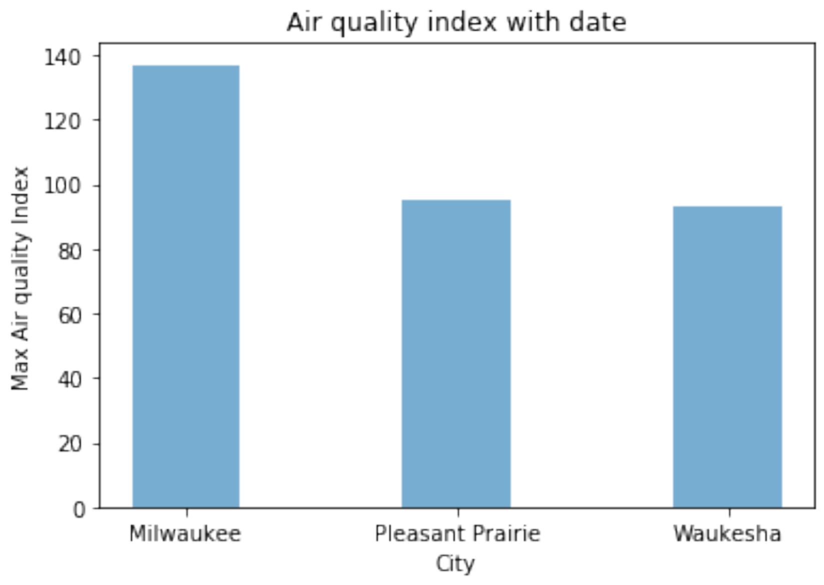
	max_aqi	city	date
0	137	Milwaukee	1997-04-04
1	95	Pleasant Prairie	1997-10-07
2	93	Waukesha	1997-04-28

Step 3 : Data Visualization

Visualize the data you received above as plots

In [9]:
```
# Use matplotlib library to plot graph - a simple barchart in this cas
e
# We encourage you to explore different and more complex types of char
t
import matplotlib.pyplot as plt


# First we convert the Dataframe object to Python list (that Matplotli
b understands)
from pandas import DataFrame
# this is our 'y-axis' value - obtained by flattening the DataFrame ob
ject
aqis_list = list(max_air_quality['max_aqi'].values.flatten())
# This is our x-axis value
cities = list(max_air_quality['city'].values.flatten())
dates = list(max_air_quality['date'].values.flatten())
# Specify the plot type and formatting
plt.title('Air quality index with date')
y_pos = np.arange(len(cities))
# Type of graph
plt.bar(y_pos, aqis_list, align='center', alpha=0.6, width=0.4)
# Specify 'ticks' on the x-axis for each item in the list for 'x-axis'
values
plt.xticks(y_pos, cities)
plt.xlabel('City')
plt.ylabel('Max Air quality Index')
plt.show()
# Save this diagram(by default at the directory from where this applic
ation is launched)
plt.savefig("max_air_quality.png")
```



Step 4 : Optional (recommended) : Build Machine Learning models

```
In [10]: from sklearn.model_selection import cross_val_predict
         from sklearn import linear_model
         import matplotlib.pyplot as plt
         df = pd.read_csv('AQI.csv')
         df = df.dropna()
         lr = linear_model.LinearRegression()
         x = df['Latitude']
         y = df.AQI

         # cross_val_predict returns an array of the same size as `y` where eac
         h entry
         # is a prediction obtained by cross validation:
         predicted = cross_val_predict(lr, x, y, cv=10)

         fig, ax = plt.subplots()
         ax.scatter(y, predicted)
         ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)
         ax.set_xlabel('Date')
         ax.set_ylabel('AQI')
         plt.show()
```

```
/Users/yuhanliu/anaconda/lib/python3.6/site-packages/IPython/core/in
teractiveshell.py:2717: DtypeWarning: Columns (0) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/utils/v
alidation.py:395: DeprecationWarning: Passing 1d arrays as data is d
eprecated in 0.17 and will raise ValueError in 0.19. Reshape your da
ta either using X.reshape(-1, 1) if your data has a single feature o
r X.reshape(1, -1) if it contains a single sample.
  DeprecationWarning)

---------------------------------------------------------------------
-------
ValueError                                Traceback (most recent cal
l last)
<ipython-input-10-d2a04705c345> in <module>()
     10 # cross_val_predict returns an array of the same size as `y`
where each entry
     11 # is a prediction obtained by cross validation:
---> 12 predicted = cross_val_predict(lr, x, y, cv=10)
     13
     14 fig, ax = plt.subplots()

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/model_s
election/_validation.py in cross_val_predict(estimator, X, y, groups
, cv, n_jobs, verbose, fit_params, pre_dispatch, method)
    399     prediction_blocks = parallel(delayed(_fit_and_predict)(
    400         clone(estimator), X, y, train, test, verbose, fit_pa
rams, method)
--> 401         for train, test in cv_iter)
    402 
    403     # Concatenate the predictions

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/externa
ls/joblib/parallel.py in __call__(self, iterable)
    756             # was dispatched. In particular this covers the
edge
    757             # case of Parallel used with an exhausted iterat
or.
--> 758             while self.dispatch_one_batch(iterator):
    759                 self._iterating = True
    760             else:

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/externa
ls/joblib/parallel.py in dispatch_one_batch(self, iterator)
    606                 return False
    607             else:
--> 608                 self._dispatch(tasks)
    609                 return True
    610 

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/externa
ls/joblib/parallel.py in _dispatch(self, batch)
    569         dispatch_timestamp = time.time()
    570         cb = BatchCompletionCallBack(dispatch_timestamp, len
(batch), self)
--> 571         job = self._backend.apply_async(batch, callback=cb)
    572         self._jobs.append(job)
    573 

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/externa
ls/joblib/_parallel_backends.py in apply_async(self, func, callback)
    107     def apply_async(self, func, callback=None):
    108         """Schedule a func to be run"""
--> 109         result = ImmediateResult(func)
    110         if callback:
    111             callback(result)

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/externa
ls/joblib/_parallel_backends.py in __init__(self, batch)
    324         # Don't delay the application, to avoid keeping the
input
    325         # arguments in memory
--> 326         self.results = batch()
    327 
    328     def get(self):

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/externa
ls/joblib/parallel.py in __call__(self)
    129
    130     def __call__(self):
--> 131         return [func(*args, **kwargs) for func, args, kwargs
in self.items]
    132
    133     def __len__(self):

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/externa
ls/joblib/parallel.py in <listcomp>(.0)
    129
    130     def __call__(self):
--> 131         return [func(*args, **kwargs) for func, args, kwargs
in self.items]
    132
    133     def __len__(self):

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/model_s
election/_validation.py in _fit_and_predict(estimator, X, y, train,
test, verbose, fit_params, method)
    472         estimator.fit(X_train, **fit_params)
    473     else:
--> 474         estimator.fit(X_train, y_train, **fit_params)
    475     func = getattr(estimator, method)
    476     predictions = func(X_test)

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/linear_
model/base.py in fit(self, X, y, sample_weight)
    510         n_jobs_ = self.n_jobs
    511         X, y = check_X_y(X, y, accept_sparse=['csr', 'csc',
'coo'],
--> 512                          y_numeric=True, multi_output=True)
    513
    514         if sample_weight is not None and np.atleast_1d(sampl
e_weight).ndim > 1:

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/utils/v
alidation.py in check_X_y(X, y, accept_sparse, dtype, order, copy, f
orce_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_sampl
es, ensure_min_features, y_numeric, warn_on_dtype, estimator)
    529         y = y.astype(np.float64)
    530
--> 531     check_consistent_length(X, y)
    532
    533     return X, y

/Users/yuhanliu/anaconda/lib/python3.6/site-packages/sklearn/utils/v
```

```
alidation.py in check_consistent_length(*arrays)
    179     if len(uniques) > 1:
    180         raise ValueError("Found input variables with inconsi
stent numbers of"
--> 181                          " samples: %r" % [int(l) for l in l
engths])
    182
    183

ValueError: Found input variables with inconsistent numbers of sampl
es: [1, 2622025]
```

Summary

1. Write a brief summary of your findings here.

2. Additionally explore by building a machine learning model using scikit-learn to make predictions and analysis

Finally, take a moment to reflect on what we achieved here - we started with a raw dataset, processed them, built models and ran analytics to finally gain insights from the data :)