# WEATHER FORECASTING

## Importing some important utilities!

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** os

**for** dirname, \_, filenames **in** os**.**walk('/kaggle/input'):

**for** filename **in** filenames:

print(os**.**path**.**join(dirname, filename))

* /kaggle/input/weather-prediction/seattle-weather.csv

df**=**pd**.**read\_csv("../input/weather-prediction/seattle-weather.csv")

Checking whether the dataset got imported to the notebook.

df**.**shape

* (1461, 6)

df**.**head()

|  | **date** | **precipitation** | **temp\_max** | **temp\_min** | **wind** | **weather** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 2012-01-01 | 0.0 | 12.8 | 5.0 | 4.7 | drizzle |
| **1** | 2012-01-02 | 10.9 | 10.6 | 2.8 | 4.5 | rain |
| **2** | 2012-01-03 | 0.8 | 11.7 | 7.2 | 2.3 | rain |
| **3** | 2012-01-04 | 20.3 | 12.2 | 5.6 | 4.7 | rain |
| **4** | 2012-01-05 | 1.3 | 8.9 | 2.8 | 6.1 | rain |

df**.**describe()

|  | **precipitation** | **temp\_max** | **temp\_min** | **wind** |
| --- | --- | --- | --- | --- |
| **count** | 1461.000000 | 1461.000000 | 1461.000000 | 1461.000000 |
| **mean** | 3.029432 | 16.439083 | 8.234771 | 3.241136 |
| **std** | 6.680194 | 7.349758 | 5.023004 | 1.437825 |
| **min** | 0.000000 | -1.600000 | -7.100000 | 0.400000 |
| **25%** | 0.000000 | 10.600000 | 4.400000 | 2.200000 |
| **50%** | 0.000000 | 15.600000 | 8.300000 | 3.000000 |
| **75%** | 2.800000 | 22.200000 | 12.200000 | 4.000000 |
| **max** | 55.900000 | 35.600000 | 18.300000 | 9.500000 |

df**.**info()

* <class 'pandas.core.frame.DataFrame'>

RangeIndex: 1461 entries, 0 to 1460

Data columns (total 6 columns):

# Column Non-Null Count Dtype

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

0 date 1461 non-null object

1 precipitation 1461 non-null float64

2 temp\_max 1461 non-null float64

3 temp\_min 1461 non-null float64

4 wind 1461 non-null float64

5 weather 1461 non-null object

dtypes: float64(4), object(2)

memory usage: 68.6+ KB

df**.**isnull()**.**sum()

* date 0

precipitation 0

temp\_max 0

temp\_min 0

wind 0

weather 0

dtype: int64

There are no null values in the data set.

## Data Pre-Processing:

The column weather contains the data value in the string form and we need to predict the weather data so for that we need to process the data.

**from** sklearn.preprocessing **import** LabelEncoder

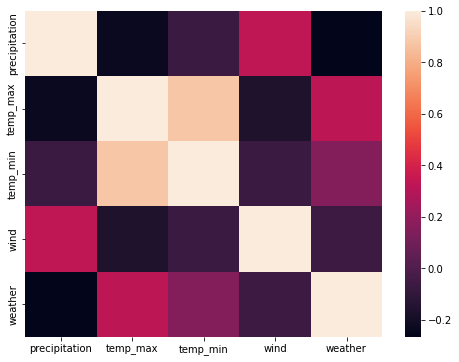
df['weather']**=**LabelEncoder()**.**fit\_transform(df['weather'])

## Data Visualization:

fig, axes**=**plt**.**subplots(figsize**=**(8, 6))

sns**.**heatmap(df**.**corr(), ax**=**axes)

* <AxesSubplot:>



We can observe that wind and preception are weakly co-related.temp\_max and wind are negatively correlated means they move in the opposite direction.

## Machine Learning Model for Prediction of Weather

**from** sklearn.model\_selection **import** train\_test\_split

features**=**["precipitation", "temp\_max", "temp\_min", "wind"]

X**=**df[features]

y**=**df**.**weather

train\_X, test\_X, train\_y, test\_y **=** train\_test\_split(X, y,random\_state **=** 0)

### Decision Tree Regressor:

**from** sklearn.tree **import** DecisionTreeRegressor

**from** sklearn.metrics **import** mean\_absolute\_error

model1**=**DecisionTreeRegressor(random\_state**=**1)

model1**.**fit(train\_X, train\_y)

pred1**=**model1**.**predict(test\_X)

print("Mean Absolute Error: %f" **%**(mean\_absolute\_error(test\_y, pred1)))

* Mean Absolute Error: 0.713115

### Random Forest Regressor:

**from** sklearn.ensemble **import** RandomForestRegressor

model2**=**RandomForestRegressor(random\_state**=**1)

model2**.**fit(train\_X, train\_y)

pred2**=**model2**.**predict(test\_X)

print("Mean Absolute Error: %f" **%**(mean\_absolute\_error(test\_y, pred2)))

* Mean Absolute Error: 0.694026

### Extreme Gradient Regressor:

**from** xgboost **import** XGBRegressor

model3**=** XGBRegressor(n\_estimators**=**100, learning\_rate**=**0.04)

model3**.**fit(train\_X, train\_y)

pred3**=**model3**.**predict(test\_X)

print("Mean Absolute Error: %f" **%**(mean\_absolute\_error(test\_y, pred3)))

* Mean Absolute Error: 0.682803

The least mean absolute error we got is of the XGB Regressor Model.