# Analysis

November 23, 2023

[ ]:
```
import pandas as pd
import numpy as np

df_original = pd.read_csv('Cleaned.csv')

columns = [
    'International ID',
    'Storm Names',
    'Date and Time',
    'Grade',
    'Latitude of the Center',
    'Longitude of the Center',
    'Central Pressure',
    'Maximum sustained wind speed',
    'Direction of the longest radius of 50kt winds or greater',
    'Longest radius of 50kt winds or greater',
    'Shortest radius of 50kt winds or greater',
    'Direction of the longest radius of 30kt winds or greater',
    'Longest radius of 30kt winds or greater'
]

na_count = df_original[columns].isna().sum()

columns_to_drop = [
    'Maximum sustained wind speed',
    'Direction of the longest radius of 50kt winds or greater',
    'Longest radius of 50kt winds or greater',
    'Shortest radius of 50kt winds or greater',
    'Direction of the longest radius of 30kt winds or greater',
    'Longest radius of 30kt winds or greater'
]

df = df_original.drop(columns_to_drop, axis =1)
df = df.drop('Unnamed: 0', axis=1)
```

```
/tmp/ipykernel_23189/1691547180.py:4: DtypeWarning: Columns (2) have mixed
types. Specify dtype option on import or set low_memory=False.
  df_original = pd.read_csv('Cleaned.csv')
```

[ ]:
```
print(type(df.loc[10,'Date and Time']))
```

```
<class 'str'>
```

[ ]:
```
import streamlit as st

td = len(df[df.Grade == 'Tropical Depression (TD)'].count(axis=1))
ts = len(df[df.Grade == 'Tropical Storm (TS)'].count(axis=1))
sts = len(df[df.Grade == 'Severe Tropical Storm (STS)'].count(axis=1))
ty = len(df[df.Grade == 'Typhoon (TY)'].count(axis=1))
l = len(df[df.Grade == 'Extra-tropical Cyclone (L)'].count(axis=1))

grade_ocurrence = [td, ts, sts, ty, l]

grade = ['Tropical Depression (TD)',
    'Tropical Storm (TS)',
    'Severe Tropical Storm (STS)',
    'Typhoon (TY)',
    'Extra-tropical Cyclone (L)']

data_ocurrence = pd.DataFrame(grade_ocurrence,grade)

st.bar_chart(data_ocurrence)
```

```
2023-10-26 10:40:46.281
  Warning: to view this Streamlit app on a browser, run it with the
following
  command:

    streamlit run /home/hub/Documents/Typhoons/typhoon-env/lib/python3.8/site-
packages/ipykernel_launcher.py [ARGUMENTS]
```

[ ]:
```
DeltaGenerator()
```

[ ]:
```
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

columns_to_use_for_training = [
    'Date and Time',
    'Latitude of the Center',
    'Longitude of the Center'
]

y = df.loc[:, 'Grade']
x = df.loc[:, columns_to_use_for_training]
```

```
train_X, val_X, train_y, val_y = train_test_split(x, y, random_state = 0)

# Applying Random Forest model
forest_model = RandomForestRegressor(random_state=1)
forest_model.fit(train_X, train_y)

# Checking the model's MAE
y_prediction = forest_model.predict(val_X)
print("The model's mean absolute error is:")
print(mean_absolute_error(val_y, y_prediction))

# Checking model's predictions
prediction = forest_model.predict(X)
difference = y-prediction
print("The predictions are:")
print(prediction)

# Checking prediction's standard deviation
print("The standard deviation associated to the predicted values is:")
print(np.std(prediction))

# Checking mean difference between true and predicted values
print("The mean difference between the actual values and predicted values is:")
print(difference.mean())
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       /tmp/ipykernel_23189/2020224142.py in ?()
   ---> 18 from sklearn.metrics import mean_absolute_error
        19 from sklearn.model_selection import train_test_split
        20 from sklearn.ensemble import RandomForestRegressor
        21


       ~/Documents/Typhoons/typhoon-env/lib/python3.8/site-packages/sklearn/
↪base.py in ?(estimator, *args, **kwargs)
      1148                 skip_parameter_validation=(
      1149                     prefer_skip_nested_validation or␣
↪global_skip_validation
      1150                 )
      1151             ):
   -> 1152                 return fit_method(estimator, *args, **kwargs)
```

```
      ~/Documents/Typhoons/typhoon-env/lib/python3.8/site-packages/sklearn/
↪ensemble/_forest.py in ?(self, X, y, sample_weight)
      344         """
      345         # Validate or convert input data
      346         if issparse(y):
      347             raise ValueError("sparse multilabel-indicator for y is␣
↪not supported.")
   --> 348         X, y = self._validate_data(
      349             X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
      350         )
      351         if sample_weight is not None:


      ~/Documents/Typhoons/typhoon-env/lib/python3.8/site-packages/sklearn/
↪base.py in ?(self, X, y, reset, validate_separately, cast_to_ndarray,␣
↪**check_params)
      618                 if "estimator" not in check_y_params:
      619                     check_y_params = {**default_check_params,␣
↪**check_y_params}
      620                 y = check_array(y, input_name="y", **check_y_params)
      621             else:
   --> 622                 X, y = check_X_y(X, y, **check_params)
      623             out = X, y
      624
      625         if not no_val_X and check_params.get("ensure_2d", True):


      ~/Documents/Typhoons/typhoon-env/lib/python3.8/site-packages/sklearn/
↪utils/validation.py in ?(X, y, accept_sparse, accept_large_sparse, dtype,␣
↪order, copy, force_all_finite, ensure_2d, allow_nd, multi_output,␣
↪ensure_min_samples, ensure_min_features, y_numeric, estimator)
     1142         raise ValueError(
     1143             f"{estimator_name} requires y to be passed, but the␣
↪target y is None"
     1144         )
     1145
   -> 1146     X = check_array(
     1147         X,
     1148         accept_sparse=accept_sparse,
     1149         accept_large_sparse=accept_large_sparse,
```

```
       ~/Documents/Typhoons/typhoon-env/lib/python3.8/site-packages/sklearn/
↪utils/validation.py in ?(array, accept_sparse, accept_large_sparse, dtype,␣
↪order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
↪ensure_min_features, estimator, input_name)
       913                     array = xp.astype(array, dtype, copy=False)
       914                 else:
       915                     array = _asarray_with_order(array, order=order,␣
↪dtype=dtype, xp=xp)
       916             except ComplexWarning as complex_warning:
   --> 917                 raise ValueError(
       918                     "Complex data not supported\n{}\n".format(array)
       919                 ) from complex_warning
       920


       ~/Documents/Typhoons/typhoon-env/lib/python3.8/site-packages/sklearn/
↪utils/_array_api.py in ?(array, dtype, order, copy, xp)
       376         # Use NumPy API to support order
       377         if copy is True:
       378             array = numpy.array(array, order=order, dtype=dtype)
       379         else:
   --> 380             array = numpy.asarray(array, order=order, dtype=dtype)
       381
       382         # At this point array is a NumPy ndarray. We convert it to␣
↪an array
       383         # container that is consistent with the input's namespace.


       ~/Documents/Typhoons/typhoon-env/lib/python3.8/site-packages/pandas/core/
↪generic.py in ?(self, dtype)
      1996     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.
↪ndarray:
      1997         values = self._values
   -> 1998         arr = np.asarray(values, dtype=dtype)
      1999         if (
      2000             astype_is_view(values.dtype, arr.dtype)
      2001             and using_copy_on_write()


       ValueError: could not convert string to float: '1963-07-01 18:00:00'
```

[ ]: