Minions

Group Members:

M Varshith

Dileepan S K

Emil Ben

Aim

- to develop a Machine Learning Model which can take Sequential Data and Generate the Sea Surface Temperature. SeaSurface Temperature is one of the important factor in prediction of El Niño.

Models Tested

- Linear Regression
- KNN
- Random Forest
- Gradient Boosting
- LSTM
- VAR
- Facebook Prophet
- Cat Boost (Our Best Model)
- XG Boost

Data Understanding:

```
3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104409 entries, 0 to 104408
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   year_month_day  104409 non-null  datetime64[ns]
 1   year            104409 non-null  object
 2   month           104409 non-null  object
 3   day             104409 non-null  object
 4   latitude        104409 non-null  float64
 5   longitude       104409 non-null  float64
 6   zon.winds       89839 non-null   float64
 7   mer.winds       89839 non-null   float64
 8   humidity        65615 non-null   float64
 9   air temp.       96571 non-null   float64
 10  s.s.temp.       104409 non-null  float64
dtypes: datetime64[ns](1), float64(7), object(3)
memory usage: 8.8+ MB
```

First and foremost the data has a lot of missing values that is when we arranged the data in terms of yy/mm/dd the humidity column is filled with null values from the year 1980-1990; only after the year 1991 the data actually had humidity filled with adequate amount of data

If we were to used this data to train the model, the data having a column with 10 years worth of null values would be not beneficial while training the data if better to use the rest of the year(1992 to 1997) filled with humidity to train our ML Models

Or else we can first try not using 10 years of data and also with all the years of data and compare the models accordingly.

//use can read the sorted data in sorted_train file our data-visualization folder//

Correlation Matrix:

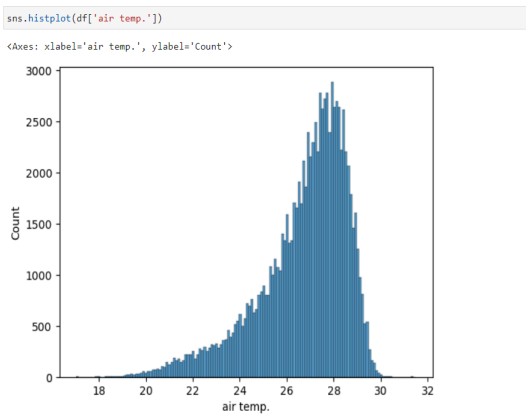
	year_month_day	year	month	day	latitude	longitude	zon.winds	mer.winds	humidity	air temp.	s.s.temp.
year_month_day	1.000000	0.995176	0.048525	0.006929	-0.002313	-0.014011	-0.040379	-0.170241	-0.064860	0.198425	0.209926
year	0.995176	1.000000	-0.049355	-0.002275	-0.002274	-0.013361	-0.045173	-0.192424	-0.037977	0.212008	0.219676
month	0.048525	-0.049355	1.000000	0.010564	-0.000169	-0.006366	0.048058	0.225312	-0.140442	-0.137345	-0.098703
day	0.006929	-0.002275	0.010564	1.000000	-0.002158	-0.002984	0.009165	0.003823	-0.004931	-0.005773	-0.003908
latitude	-0.002313	-0.002274	-0.000169	-0.002158	1.000000	0.039898	0.127751	0.002152	0.169732	0.070133	0.127508
longitude	-0.014011	-0.013361	-0.006366	-0.002984	0.039898	1.000000	0.419216	-0.118610	-0.039895	0.350036	0.425969
zon.winds	-0.040379	-0.045173	0.048058	0.009165	0.127751	0.419216	1.000000	0.071219	0.062317	0.210095	0.342753
mer.winds	-0.170241	-0.192424	0.225312	0.003823	0.002152	-0.118610	0.071219	1.000000	0.120932	-0.385561	-0.334296
humidity	-0.064860	-0.037977	-0.140442	-0.004931	0.169732	-0.039895	0.062317	0.120932	1.000000	-0.409888	-0.355206
air temp.	0.198425	0.212008	-0.137345	-0.005773	0.070133	0.350036	0.210095	-0.385561	-0.409888	1.000000	0.949209
s.s.temp.	0.209926	0.219676	-0.098703	-0.003908	0.127508	0.425969	0.342753	-0.334296	-0.355206	0.949209	1.000000

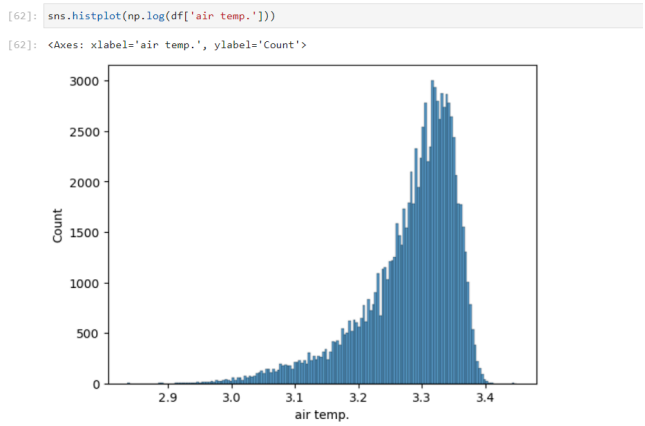
As you can see from the above from the above correlation matrix the correlation of s.s.temperature with air temp is very high and because of that we cannot blindly fill the null values associated with the air temperature.More about this later

Distribution Plot:

Some of the columns(air temp,zon temp,) are left distributed indicating the data is skewed.

Usually we don't want our data to be skewed as the model might find it hard to read the data and create the algorithm and it is usually removed by using log transformation which converts into a normal distribution,but as you can see (pic on left after log trans)





Even after applying log transformation the distribution is still left distributed;

To account for did we used standard scaler as it seems the be a better approach after test out a lot of models and also it also helping with fixing outlier in our data

Box Plot:

Columns with outliers: humidity,air temp,mer winds,zon wind and s.s.temp

As we Mentioned before Standard Scaler seems to take care of the data filled with outlier better than MinMaxScaler

Filling the Null Values:

As we mentioned before we dropped the rows containing air temp Null value due to its high correlation with s.s.temp as filling it with random value(also it has no good correlation with other features) will make our model inaccurate.

For the rest of the features containing null values; after using various models normal distribution seems to give a better output than filling it with median or mean

Spilitting our data:

Since this is a time series data it is always important to sort our data(in order of time) and split it using time series splitting(library) and what this time series slitting does is it trains our model (lets say we have data from 1992-1997) from 1992 -1995 and test the data on 1995-1997 since we are technically forecasting our data we have to make sure our future predictions are accurate.This is a crucial step in our opinion.

Models:

Before we talk about Models and their implementations

an import thing about this data

Data as a Time Series:

Literal time series definition is that it is a collection of data points gathered over a period of time and ordered chronologically.But in a way this time series data is bad data for time series model.Let me explain.

There are certain specific time series algorithms specified to this task for example :

Univiariate Algorithms:

ARIMA

Multivariate Algorithms(our data):

Vector Auto Regression (VAR) ,Lstm , Facebook Prophet

Usually these algorithms are used forecasting because these have a little edge over other algorithms.But these have a fatal flaw i.e. these algorithms don't account for spatial coordinates or in simpler word these algorithm assumes our point is a stationarity point, but our data contains a lot of data points on the same day ;like say on 1987 march 14 we have 6 data points indifferent places and some the frequence of days also varies along with location sometimes its 3 days of different location

1987	3	14	-2.01	-110.0	-1.9	3.0	
1987	3	14	-4.96	165.0	-0.8	0.5	
1987	3	14	-0.09	165.1	-1.6	-1.7	
1987	3	14	2.0	165.0	-2.2	-2.0	
1987	3	14	-8.07	-109.94	-2.8	4.1	
1987	3	14	0.0	-109.94	-3.8	3.4	

This inconsistency in our data affects our time series algorithms/models very tremendously

But there is a way to mitigate this huge variability in our location by binning the longitude and latitude into lets say 5 or 6 bins and making it into a categorical feature and one hot encoding it but

```
]: df[['longitude','latitude']].nunique()

]: longitude    355
   latitude     608
   dtype: int64
```

With these many unique values in our data its kind of complicated to bin the data and the complexity might increase with data.

So we thought its better to use regression(in a sense models other than time series one's) models than building time series models (but we build them anyway cus we had a lot of time) and our outputs were significantly better than our time series models.

Coming back to our models and performances

Linear Regression:

```
mean_absolute_error(y_test,pred)
```

```
0.43088856791121427
```

```
root_mean_squared_error(y_test,pred)
```

```
0.5694029944459054
```

```
r2_score(y_test,pred)
```

```
0.9397168830708291
```

KNN:

```
mean_absolute_error(y_test,pred)
```

```
0.4440597299312402
```

```
root_mean_squared_error(y_test,pred)
```

```
0.5693781117860918
```

```
r2_score(y_test,pred)
```

```
0.9397221516469727
```

Random Forest:

```
mean_absolute_error(y_test,pred)
```

```
0.3880456383066856
```

```
root_mean_squared_error(y_test,pred)
```

```
0.5053097286104328
```

```
r2_score(y_test,pred)
```

```
0.9525242837781187
```

Gradient Boosting:

```
mean_absolute_error(y_test,pred)
```

```
0.3952731218112942
```

```
root_mean_squared_error(y_test,pred)
```

```
0.5057111075729986
```

```
r2_score(y_test,pred)
```

```
0.9524488317499734
```

XG Boost:

```
Mean Squared Error: 0.2834742251011486
R-squared: 0.9449664548944016
Model Score (R²): 0.9449664548944016
       Actual  Predicted
```

Cat Boost:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test,y_pred)
```

```
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
print(f"Mean Absolute Error: {mae}")
```

```
Mean Squared Error: 0.23857326367862705
R-squared: 0.9556413850186
Mean Absolute Error: 0.3778479986520532
```

```
model_score = catboost_model.score(x_test, y_test)
print(f"Model Score (R²): {model_score}")
```

```
Model Score (R²): 0.9556413850186
```

Time Series Models:

LSTM:

```
451/451 [==============================] - 12s 25ms/step
Mean Squared Error: 4.949178533776575
```

VAR:

Tested the predictions they were way off

Facebook Prophet:

```
Mean Absolute Error: 1.75543504326188
Root Mean Squared Error: 2.2458428567001305
```

Model Evaluation:

Model evaluation while evaluating the model we have tried many other methods to optimize the model like using pipeline,grid search,min-max scaler and also looked for the data whether is over fits or under fits the data and came to the final conclusion that CatBoost Regression gave the best results with our data