Model Definition and Model Training

This Notebook is for Non-Normalised Dataset

Data Dictionary

Field	Description
Date	Date of incident
Province/State	If there is a breakdown in states
Lat	Latitude
Lon	Longitude
Confirmed	Confirmed cases
Recovered	Recovered cases
Deaths	Death cases
Active	Active cases

Summary and Results

Model performance indicator will be Root Mean Square Error (RMSE) to be as accurate as possible.

Results

Algorithm	MSE	RMSE	R2
Linear Regression	2.038	1.428	0.319
Extra Trees Regressor	2.282	1.511	0.238
XGBoost Regressor	1.656	1.287	0.447

Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import shap

import xgboost as xgb
from xgboost import XGBRegressor
from xgboost import plot_importance

%matplotlib inline
sns.set_style('dark')
sns.set(font_scale=1.2)

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import ExtraTreesRegressor

from sklearn.pipeline import Pipeline
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.feature_selection import RFE, RFECV, SelectKBest, f_classif, f_regression, chi2

from sklearn.inspection import permutation_importance
from sklearn.model_selection import cross_val_score, train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix, classification_report, mean_absolute_error, mean_squared_error
from sklearn.metrics import plot_confusion_matrix, plot_precision_recall_curve, plot_roc_curve, accuracy_sco
from sklearn.metrics import auc, f1_score, precision_score, recall_score, roc_auc_score

import warnings
warnings.filterwarnings('ignore')

import pickle
from pickle import dump, load

np.random.seed(0)

from pycaret.regression import *

pd.set_option('display.max_columns',50)
#pd.set_option('display.max_rows',100)
pd.set_option('display.width', 1000)
np.set_printoptions(suppress=True)
```

Data Exploration and Analysis

```
df = pd.read_csv("malaysiacovid2.csv")
```

```
df.head()
```

	Date	Confirmed	Recovered	Deaths	Active	ConfirmDiff	DeathsDiff	RecoverDiff	ActiveDiff	Month	Day
0	2020-01-22	0	0	0	0	0.0	0.0	0.0	0.0	1	22
1	2020-01-23	0	0	0	0	0.0	0.0	0.0	0.0	1	23
2	2020-01-24	0	0	0	0	0.0	0.0	0.0	0.0	1	24
3	2020-01-25	3	0	0	3	3.0	0.0	0.0	3.0	1	25
4	2020-01-26	4	0	0	4	1.0	0.0	0.0	1.0	1	26

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         309 non-null    object
 1   Confirmed    309 non-null    int64
 2   Recovered    309 non-null    int64
 3   Deaths       309 non-null    int64
 4   Active       309 non-null    int64
 5   ConfirmDiff  309 non-null    float64
 6   DeathsDiff   309 non-null    float64
 7   RecoverDiff  309 non-null    float64
 8   ActiveDiff   309 non-null    float64
 9   Month        309 non-null    int64
 10  Day          309 non-null    int64
dtypes: float64(4), int64(6), object(1)
memory usage: 26.7+ KB
```

```
df.describe(include='all')
```

	Date	Confirmed	Recovered	Deaths	Active	ConfirmDiff	DeathsDiff	RecoverDiff	ActiveDiff	Month	
count	309	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000	309.0
unique	309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	2020-11-19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	NaN	10535.511327	8236.019417	111.669903	2187.822006	193.582524	1.116505	150.488673	41.977346	6.265372	15.8
std	NaN	12082.498984	8826.720903	81.039279	3547.802649	359.375952	1.988445	306.753450	172.885226	2.943317	8.8
min	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-1382.000000	1.000000	1.0
25%	NaN	4119.000000	1487.000000	65.000000	153.000000	7.000000	0.000000	6.000000	-12.000000	4.000000	8.0
50%	NaN	8596.000000	8231.000000	121.000000	751.000000	33.000000	0.000000	28.000000	3.000000	6.000000	16.0
75%	NaN	9583.000000	9143.000000	128.000000	1973.000000	150.000000	1.000000	96.000000	51.000000	9.000000	23.0
max	NaN	59817.000000	46501.000000	345.000000	14353.000000	2188.000000	12.000000	2348.000000	1027.000000	11.000000	31.0

```
df.shape
```

```
(309, 11)
```

```
df.columns
```

```
Index(['Date', 'Confirmed', 'Recovered', 'Deaths', 'Active', 'ConfirmDiff', 'DeathsDiff', 'RecoverDiff', 'Ac
tiveDiff', 'Month', 'Day'], dtype='object')
```

Drop unwanted features

```
df.drop(['Date', 'Confirmed', 'Recovered', 'Deaths', 'Active', 'ActiveDiff'],axis=1,inplace=True)
```

```
df.head()
```

	Month	Day	ConfirmDiff	RecoverDiff	DeathsDiff
0	1	22	0.0	0.0	0.0
1	1	23	0.0	0.0	0.0
2	1	24	0.0	0.0	0.0
3	1	25	3.0	0.0	0.0
4	1	26	1.0	0.0	0.0

```
df.columns
```

```
Index(['ConfirmDiff', 'DeathsDiff', 'RecoverDiff', 'Month', 'Day'], dtype='object')
```

```
df = df[['Month', 'Day','ConfirmDiff','RecoverDiff', 'DeathsDiff']]
```

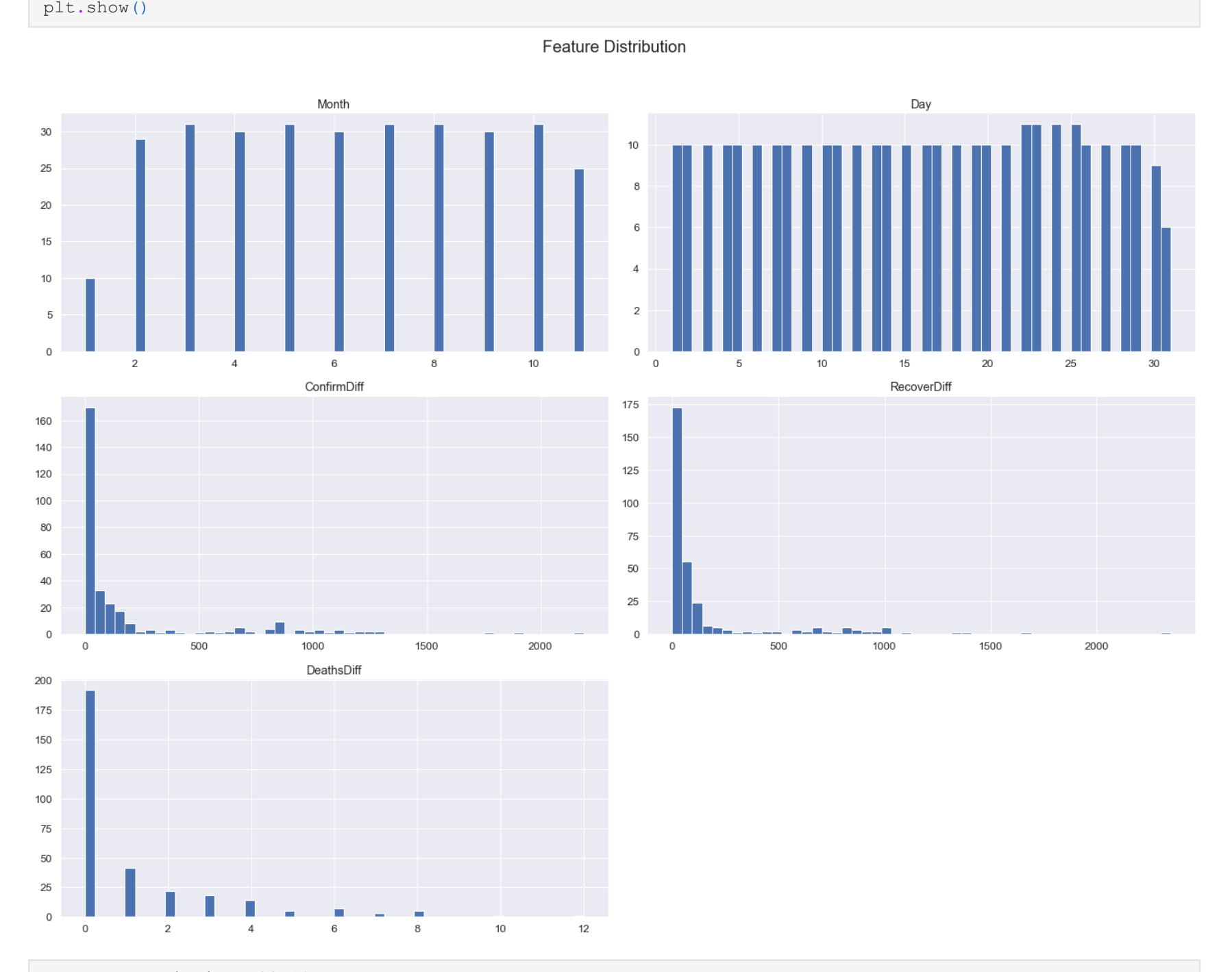
```
df.head()
```

	Month	Day	ConfirmDiff	RecoverDiff	DeathsDiff
0	1	22	0.0	0.0	0.0
1	1	23	0.0	0.0	0.0
2	1	24	0.0	0.0	0.0
3	1	25	3.0	0.0	0.0
4	1	26	1.0	0.0	0.0

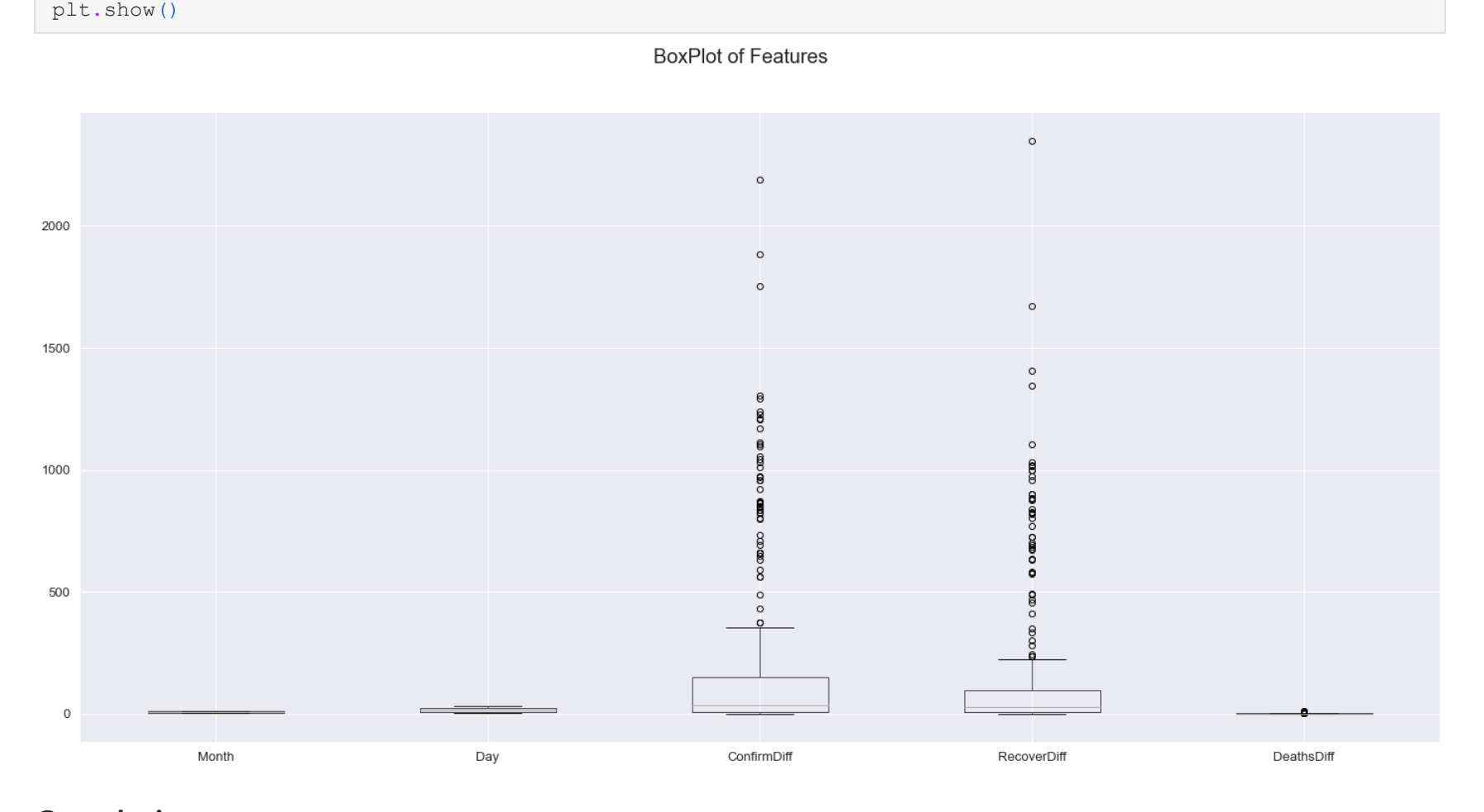
Data Visualization

Univariate Data Exploration

```
df.hist(bins=50, figsize=(20,15))
plt.suptitle('Feature Distribution', x=0.5, y=1.02, ha='center', fontsize=20)
plt.tight_layout()
plt.show()
```




```
df.boxplot(figsize=(20,10))
plt.suptitle('BoxPlot of Features', x=0.5, y=1.02, ha='center', fontsize=20)
plt.tight_layout()
plt.show()
```

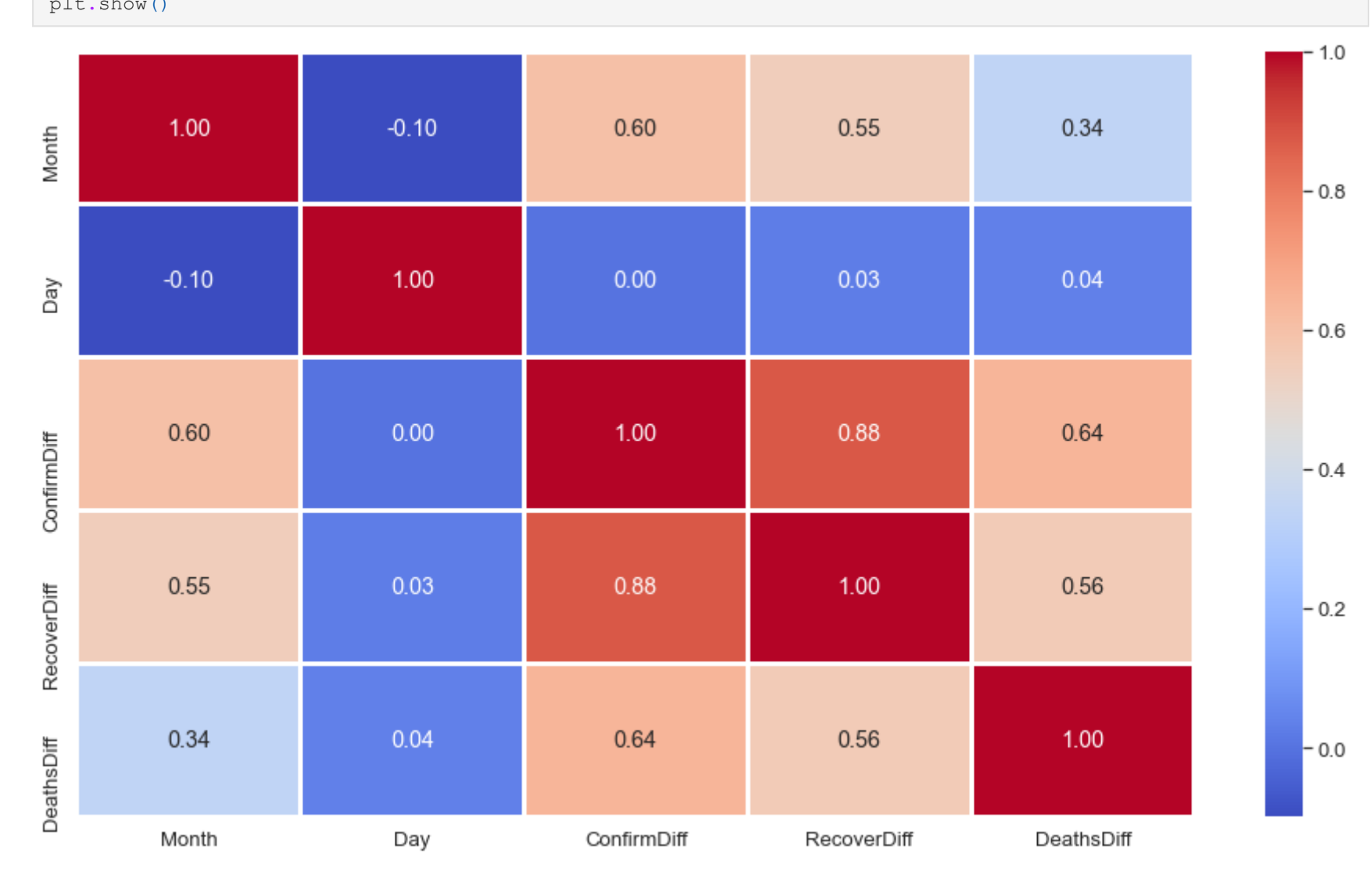


Correlation

```
df.corr()
```

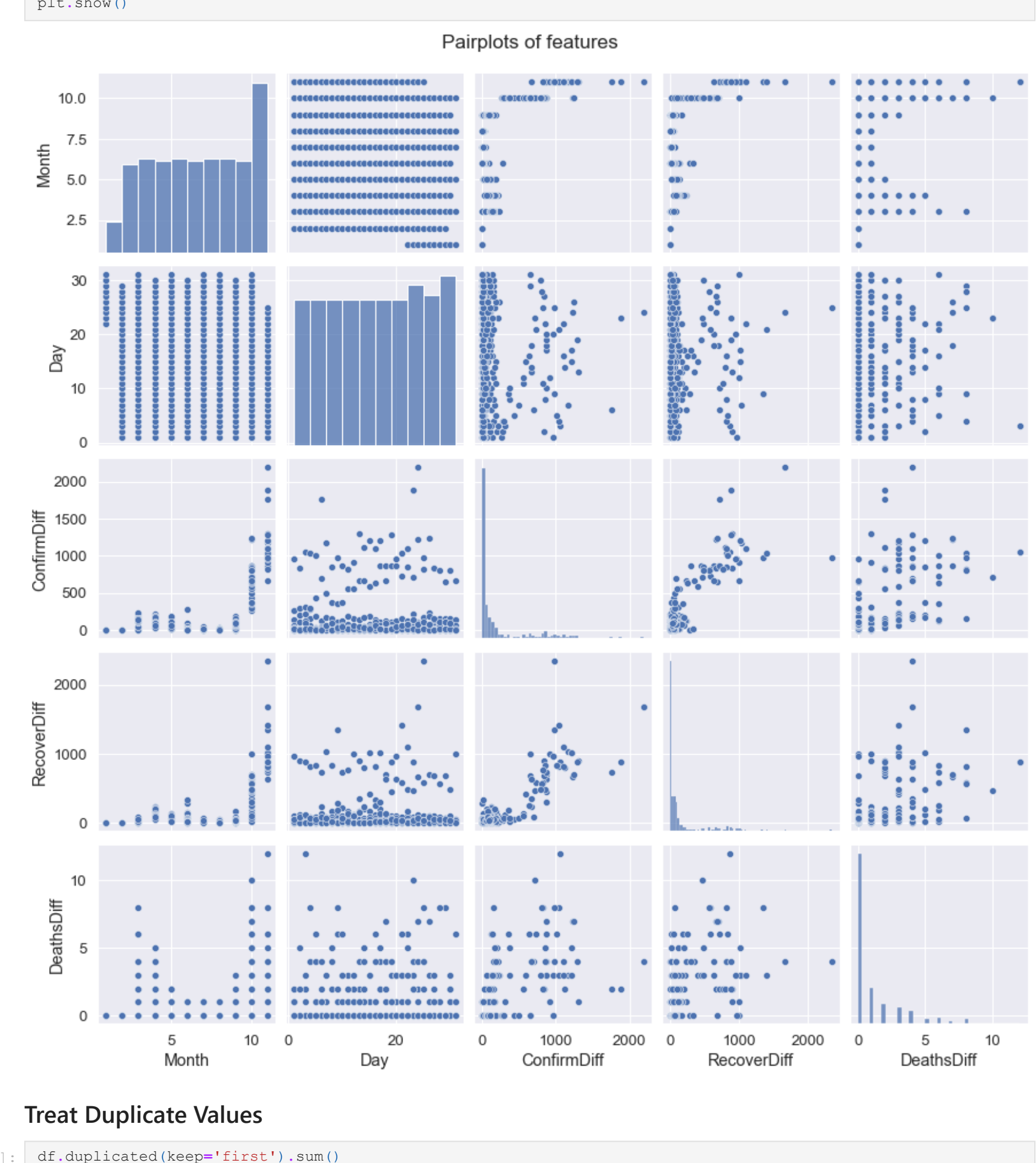
	Month	Day	ConfirmDiff	RecoverDiff	DeathsDiff
Month	1.000000	-0.098535	0.604269	0.545945	0.341420
Day	-0.098535	1.000000	0.002108	0.027499	0.035592
ConfirmDiff	0.604269	0.002108	1.000000	0.875959	0.641523
RecoverDiff	0.545945	0.027499	0.875959	1.000000	0.561145
DeathsDiff	0.341420	0.035592	0.641523	0.561145	1.000000

```
plt.figure(figsize=(16,9))
sns.heatmap(df.corr(),cmap="coolwarm",annot=True,fmt='.2f',linewidths=2)
plt.show()
```



Pairplots

```
sns.pairplot(df)
plt.suptitle('Pairplots of features', x=0.5, y=1.02, ha='center', fontsize='large')
plt.show()
```



Treat Duplicate Values

```
df.duplicated(keep='first').sum()
```

```
0
```

Treat Outliers

```
df.describe()
```

	Month	Day	ConfirmDiff	RecoverDiff	DeathsDiff
count	309.000000	309.000000	309.000000	309.000000	309.000000
mean	6.265372	15.857605	193.582524	150.488673	1.116505
std	2.943317	8.814408	359.375952	306.753450	1.988445
min	1.000000	1.000000	0.000000	0.000000	0.000000
25%	4.000000	8.000000	7.000000	6.000000	0.000000
50%	6.000000	16.000000	33.000000	28.000000	0.000000
75%	9.000000	23.000000	150.000000	96.000000	1.000000
max	11.000000	31.000000	2188.000000	2348.000000	12.000000

No treatment of outliers needed

Treat Data Types

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Month        309 non-null    int64
 1   Day          309 non-null    int64
 2   ConfirmDiff  309 non-null    float64
 3   RecoverDiff  309 non-null    float64
 4   DeathsDiff   309 non-null    float64
dtypes: float64(3), int64(2)
memory usage: 12.2 KB
```

```
df["Month"] = df["Month"].astype("category")
```

```
df["Day"] = df["Day"].astype("category")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Month        309 non-null    category
 1   Day          309 non-null    category
 2   ConfirmDiff  309 non-null    float64
 3   RecoverDiff  309 non-null    float64
 4   DeathsDiff   309 non-null    float64
dtypes: category(2), float64(3)
memory usage: 9.9 KB
```

Create and save processed dataset

```
#df.to_csv("traincovid.csv",index=False)
```

```
df["Month"].value_counts()
```

```
10    31
8     31
7     31
5     31
3     31
9     30
6     30
4     30
2     29
11    25
1     10
Name: Month, dtype: int64
```

```
df["Day"].value_counts()
```

```
23    11
25    11
24    11
22    11
6     10
11    10
10    10
9     10
8     10
7     10
16    10
13    10
5     10
4     10
3     10
2     10
12    10
1     10
14    10
15    10
17    10
18    10
19    10
20    10
21    10
26    10
27    10
28    10
29    10
30     9
31     6
Name: Day, dtype: int64
```

```
df2 = pd.get_dummies(data=df, drop_first=True)
```

```
df2.head()
```

	ConfirmDiff	RecoverDiff	DeathsDiff	Month_2	Month_3	Month_4	Month_5	Month_6	Month_7	Month_8	Month_9	Month_10	Montl
0	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	
1	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	
2	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	
3	3.0	0.0	0.0	0	0	0	0	0	0	0	0	0	
4	1.0	0.0	0.0	0	0	0	0	0	0	0	0	0	

```
df2.columns
```

```
Index(['ConfirmDiff', 'RecoverDiff', 'DeathsDiff', 'Month_2', 'Month_3', 'Month_4', 'Month_5', 'Month_6', 'M
onth_7', 'Month_8', 'Month_9', 'Month_10', 'Month_11', 'Day_2', 'Day_3', 'Day_4', 'Day_5', 'Day_6', 'Day_7',
'Day_8', 'Day_9', 'Day_10', 'Day_11', 'Day_12', 'Day_13', 'Day_14', 'Day_15', 'Day_16', 'Day_17', 'Day_18',
'Day_19', 'Day_20', 'Day_21', 'Day_22', 'Day_23', 'Day_24', 'Day_25', 'Day_26', 'Day_27', 'Day_28', 'Day_2
9', 'Day_30', 'Day_31'], dtype='object')
```

```
df2 = df2[['Month_2', 'Month_3', 'Month_4', 'Month_5', 'Month_6', 'Month_7', 'Month_8', 'Month_9',
           'Month_10', 'Month_11', 'Day_2', 'Day_3', 'Day_4', 'Day_5', 'Day_6', 'Day_7', 'Day_8', 'Day_9',
           'Day_10', 'Day_11', 'Day_12', 'Day_13', 'Day_14', 'Day_15', 'Day_16', 'Day_17', 'Day_18',
           'Day_19', 'Day_20', 'Day_21', 'Day_22', 'Day_23', 'Day_24', 'Day_25', 'Day_26', 'Day_27',
           'Day_28', 'Day_29', 'Day_30', 'Day_31', 'ConfirmDiff', 'RecoverDiff', 'DeathsDiff']]
```

```
df2.columns
```

```
Index(['Month_2', 'Month_3', 'Month_4', 'Month_5', 'Month_6', 'Month_7', 'Month_8', 'Month_9', 'Month_10',
'Month_11', 'Day_2', 'Day_3', 'Day_4', 'Day_5', 'Day_6', 'Day_7', 'Day_8', 'Day_9', 'Day_10', 'Day_11', 'Day
_12', 'Day_13', 'Day_14', 'Day_15', 'Day_16', 'Day_17', 'Day_18', 'Day_19', 'Day_20', 'Day_21', 'Day_22', 'D
ay_23', 'Day_24', 'Day_25', 'Day_26', 'Day_27', 'Day_28', 'Day_29', 'Day_30', 'Day_31', 'ConfirmDiff', 'Reco
verDiff', 'DeathsDiff'], dtype='object')
```

```
df2.head()
```

	Month_2	Month_3	Month_4	Month_5	Month_6	Month_7	Month_8	Month_9	Month_10	Month_11	Day_2	Day_3	Day_4	Day_5	Da
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Create and save processed dataset

```
#df2.to_csv("traincovid2.csv",index=False)
```

Train Test Split

```
df2.shape
```

```
(309, 43)
```

```
X = df2.iloc[:,0:42]
y = df2.iloc[:,42]
```

```
X.values, y.values
```

```
(array([[   0.,    0.,    0., ...,    0.,    0.,    0.],
        [   0.,    0.,    0., ...,    0.,    0.,    0.],
        [   0.,    0.,    0., ...,    0.,    0.,    0.],
        ...,
        [   0.,    0.,    0., ...,    0., 1884.,  883.],
        [   0.,    0.,    0., ...,    0., 2188., 1673.],
        [   0.,    0.,    0., ...,    0.,  970., 2348.]]),
 array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  2.,  0.,  0.,  1.,  1.,  6.,  4.,  2.,  4.,  3.,
         3.,  1.,  8.,  2.,  6.,  2.,  5.,  3.,  4.,  4.,  1.,  1.,  2.,
         2.,  3.,  3.,  3.,  1.,  5.,  1.,  1.,  2.,  2.,  1.,  0.,  3.,
         1.,  2.,  1.,  2.,  0.,  1.,  1.,  0.,  2.,  1.,  0.,  2.,  0.,
         1.,  1.,  0.,  0.,  1.,  0.,  1.,  0.,  2.,  1.,  0.,  1.,  0.,
         0.,  1.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  1.,  0.,  1.,
         1.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
         0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         1.,  1.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  3.,  0.,
         0.,  0.,  1.,  0.,  0.,  2.,  0.,  0.,  1.,  0.,  0.,  4.,  0.,
         5.,  6.,  3.,  2.,  2.,  4.,  4.,  3.,  6.,  4.,  7.,  3.,  3.,
         6.,  5., 10.,  7.,  8.,  7.,  2.,  8.,  0.,  3.,  0.,  0.,  2.,
        12.,  8.,  6.,  2.,  3.,  4.,  8.,  6.,  2.,  1.,  1.,  2.,  3.,
         4.,  5.,  4.,  4.,  3.,  3.,  3.,  2.,  4.,  4.]))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((231, 42), (78, 42), (231,), (78,))
```

Feature Scaling

```
#X_train
```

```
#scaler = StandardScaler()
```

```
#minmax = MinMaxScaler()
```

```
#X_train_scaled = minmax.fit_transform(X_train)
```

```
#X_test_scaled = minmax.transform(X_test)
```

```
#X_train_scaled
```

```
#X_test_scaled
```

Model Training

Using PyCaret

```
pycaret = pd.read_csv("traincovid.csv")
```

```
exp_reg = setup(data = pycaret, target = 'DeathsDiff', session_id=0, normalize=False,
                train_size=0.75, categorical_features=['Month','Day'])
```

	Description	Value
0	session_id	0
1	Target	DeathsDiff
2	Original Data	(309, 5)
3	Missing Values	False
4	Numeric Features	2
5	Categorical Features	2
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(231, 44)
10	Transformed Test Set	(78, 44)
11	Shuffle Train-Test	True
12	Stratify Train-Test	False
13	Fold Generator	KFold
14	Fold Number	10
15	CPU Jobs	-1
16	Use GPU	False
17	Log Experiment	False
18	Experiment Name	reg-default-name
19	USI	f612
20	Imputation Type	simple
21	Iterative Imputation Iteration	None
22	Numeric Imputer	mean
23	Iterative Imputation Numeric Model	None
24	Categorical Imputer	constant
25	Iterative Imputation Categorical Model	None
26	Unknown Categoricals Handling	least_frequent
27	Normalize	False
28	Normalize Method	None
29	Transformation	False
30	Transformation Method	None
31	PCA	False
32	PCA Method	None
33	PCA Components	None
34	Ignore Low Variance	False
35	Combine Rare Levels	False
36	Rare Level Threshold	None
37	Numeric Binning	False
38	Remove Outliers	False
39	Outliers Threshold	None
40	Remove Multicollinearity	False
41	Multicollinearity Threshold	None
42	Clustering	False
43	Clustering Iteration	None
44	Polynomial Features	False
45	Polynomial Degree	None
46	Trignometry Features	False
47	Polynomial Threshold	None
48	Group Features	False
49	Feature Selection	False
50	Features Selection Threshold	None
51	Feature Interaction	False
52	Feature Ratio	False
53	Interaction Threshold	None
54	Transform Target	False
55	Transform Target Method	box-cox

```
compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lasso	Lasso Regression	1.0044	2.6196	1.5108	0.2948	0.4911	0.5943	0.0130
en	Elastic Net	1.0042	2.6199	1.5109	0.2948	0.4909	0.5947	0.0120
knn	K Neighbors Regressor	0.8153	2.2315	1.4357	0.2762	0.4397	0.7076	0.0140
huber	Huber Regressor	0.8071	2.5101	1.4925	0.2610	0.4397	0.6261	0.0290
lightgbm	Light Gradient Boosting Machine	0.8245	2.0951	1.3988	0.2433	0.4450	0.6732	0.0260
br	Bayesian Ridge	0.9737	2.6649	1.5262	0.2267	0.4839	0.6006	0.0150
et	Extra Trees Regressor	0.7696	2.5556	1.5178	0.1749	0.4442	0.6348	0.1230
catboost	CatBoost Regressor	0.8570	2.3486	1.4838	0.1551	0.4441	0.6759	1.1140
rf	Random Forest Regressor	0.8110	2.3499	1.4723	0.1445	0.4474	0.6468	0.1410
ada	AdaBoost Regressor	1.0312	2.3350	1.4824	0.1330	0.5379	0.5177	0.0440
gbr	Gradient Boosting Regressor	0.8746	2.5151	1.5329	0.0866	0.4669	0.7084	0.0390
omp	Orthogonal Matching Pursuit	0.8947	2.5398	1.5172	-0.0232	0.4715	0.6665	0.0130
ridge	Ridge Regression	1.0666	2.9943	1.6623	-0.2045	0.5479	0.6564	0.0100
llar	Lasso Least Angle Regression	1.4534	4.3269	2.0016	-0.2747	0.7435	0.4629	0.0110
dt	Decision Tree Regressor	0.9196	3.4565	1.8041	-0.2892	0.5540	0.7429	0.0120
lr	Linear Regression	1.1351	3.2520	1.7385	-0.4266	0.5768	0.6966	0.6000
par	Passive Aggressive Regressor	2.7040	49.8622	4.9473	-15.6351	0.7104	1.8892	0.0130
lar	Least Angle Regression	7703.8101	154588804.2501	13018.4320	-542510564.0134	2.5913	5098.4051	0.0150

```
Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=0,
      selection='cyclic', tol=0.0001, warm_start=False)
```

Linear Regression

```
lr = LinearRegression()
```

```
model1 = lr.fit(X_train,y_train)
```

```
model1
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
lrprediction = model1.predict(X_test)
```

```
lrprediction
```

```
array([ 1.52302928,  0.52206879, -0.72297798,  0.63594962,  0.80806581,
       -0.3670357 ,  0.11115833,  0.63505182,  0.19810741,  1.9885316 ,
        2.17430917,  1.75831649, -0.5606298 ,  1.29492258, -0.28667605,
       -0.25969866, -0.0579919 ,  0.40014286, -0.0264301 , -0.118329  ,
        0.90519578,  0.6247919 ,  1.29413271,  1.1014569 , -0.1272126 ,
        1.20515671,  1.89062302,  0.32272939,  0.91263431,  1.41336108,
        0.30666462,  0.31243144,  2.98305746,  0.15288449,  0.6306367 ,
       -0.41244754,  0.70707757,  2.91743896,  0.00654834, -0.12224363,
        1.25507736,  1.35140372,  0.28827555,  0.24357661,  0.21489345,
       -0.36487875, -0.41089488,  0.83234452,  2.7627294 , -0.03140478,
        0.38029692,  4.80719401,  1.44298252, -0.80258205,  1.34335525,
        0.27719266,  0.67624502,  0.27601275,  1.08075557, -0.20336378,
       -0.51952615,  1.06713293,  0.42887705,  0.02500505,  2.31610025,
        0.51743377, -0.72666368, -0.17666269,  0.98344387,  0.09863617,
        0.15737501,  0.34253726,  1.27240564,  0.65861651,  2.43413688,
        3.46900514,  2.62969056,  0.51160967])
```

```
lrprediction.round()
```

```
array([ 2.,  1., -1.,  1.,  1., -0.,  0.,  1.,  0.,  2.,  2.,  2., -1.,
        1., -0., -0., -0.,  0., -0., -0.,  1.,  1.,  1.,  1., -0.,  1.,
        2.,  0.,  1.,  1.,  0.,  0.,  3.,  0.,  1., -0.,  1.,  3.,  0.,
       -0.,  1.,  1.,  0.,  0.,  0., -0., -0.,  1.,  3., -0.,  0.,  5.,
        1., -1.,  1.,  0.,  1.,  0.,  1., -0., -1.,  1.,  0.,  0.,  2.,
        1., -1., -0.,  1.,  0.,  0.,  0.,  1.,  1.,  2.,  3.,  3.,  1.])
```

Model Evaluation

```
mse = mean_squared_error(y_test,lrprediction.round())
mse
```

```
2.0384615384615383
```

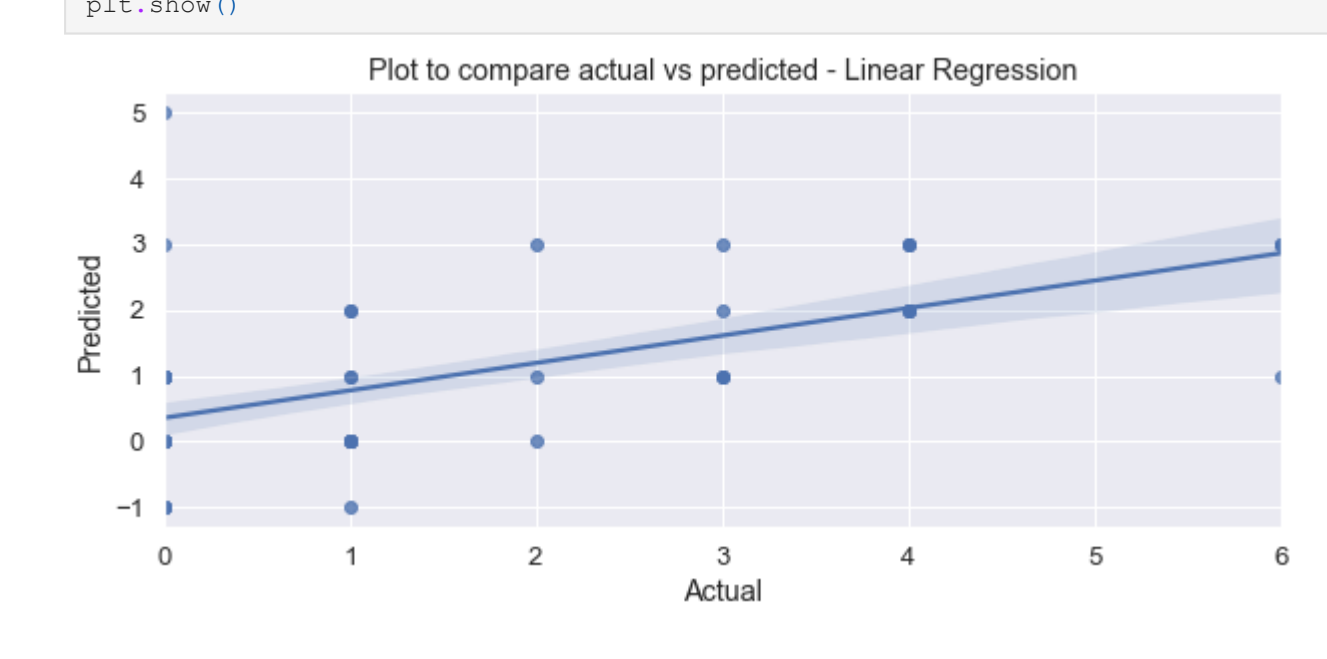
```
rmse = np.sqrt(mse)
rmse
```

```
1.427740714866968
```

```
r2score = r2_score(y_test,lrprediction.round())
r2score
```

```
0.3191699604743083
```

```
fig, ax = plt.subplots(figsize=(10,4))
sns.regplot(x=y_test, y=lrprediction.round(), ax=ax)
plt.title("Plot to compare actual vs predicted - Linear Regression")
plt.ylabel("Predicted")
plt.xlabel("Actual")
plt.show()
```



Extra Trees Regressor

```
et = ExtraTreesRegressor()
```

```
model2 = et.fit(X_train,y_train)
```

```
model2
```

```
ExtraTreesRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    max_samples=None, min_impurity_decrease=0.0,
```