INSTAGRAM REACH FORECASTING MODEL

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

```
df = pd.read_csv('Instagram-Reach.csv')
```

```
data = df
```

```
df.head()
```

	Date	Instagram reach
0	2022-04-01T00:00:00	7620
1	2022-04-02T00:00:00	12859
2	2022-04-03T00:00:00	16008
3	2022-04-04T00:00:00	24349
4	2022-04-05T00:00:00	20532

DATA CLEANING

```
df.isnull().sum()
```

```
Date              0
Instagram reach   0
dtype: int64
```

DATA ANALYSIS

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Date             365 non-null    object
 1   Instagram reach  365 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 5.8+ KB
```

```
df.shape
```

```
(365, 2)
```

```
df.describe()
```

	Instagram reach
count	365.000000
mean	50474.712329
std	30051.787552

```
df.drop_duplicates(inplace=True)
```

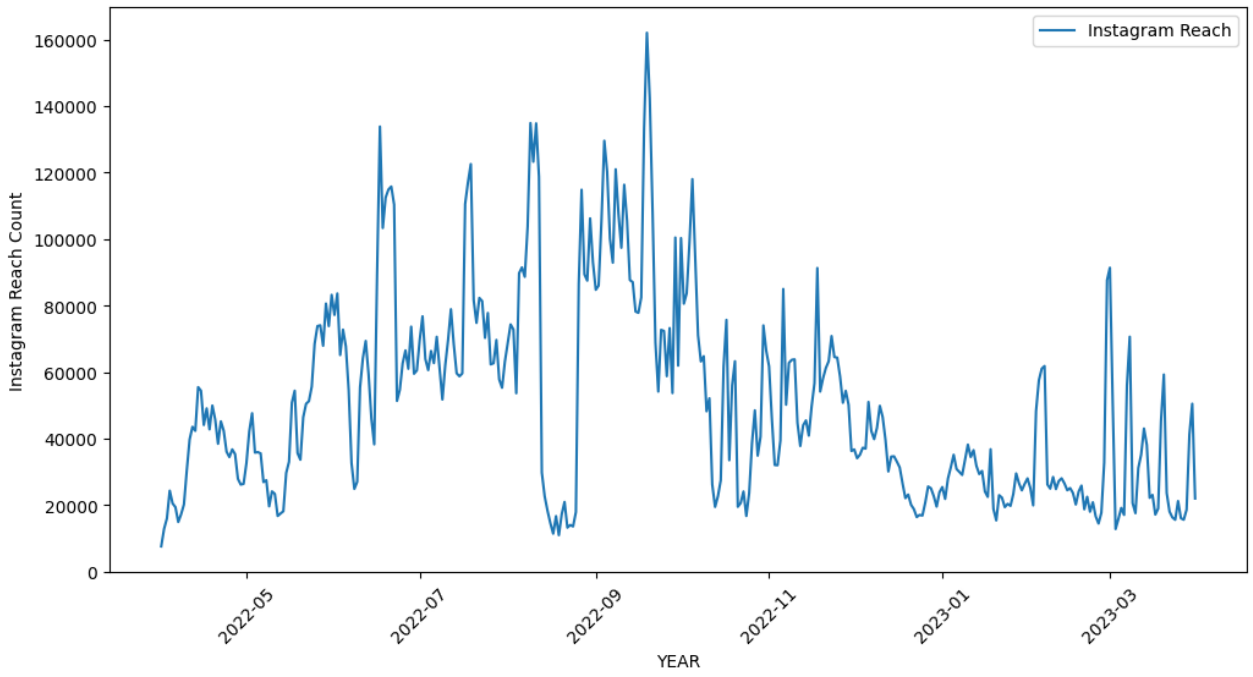
```
df.head()
```

	Date	Instagram reach
0	2022-04-01T00:00:00	7620
1	2022-04-02T00:00:00	12859
2	2022-04-03T00:00:00	16008
3	2022-04-04T00:00:00	24349
4	2022-04-05T00:00:00	20532

```
#converting date into time series
df['Date'] = pd.to_datetime(df['Date'])
```

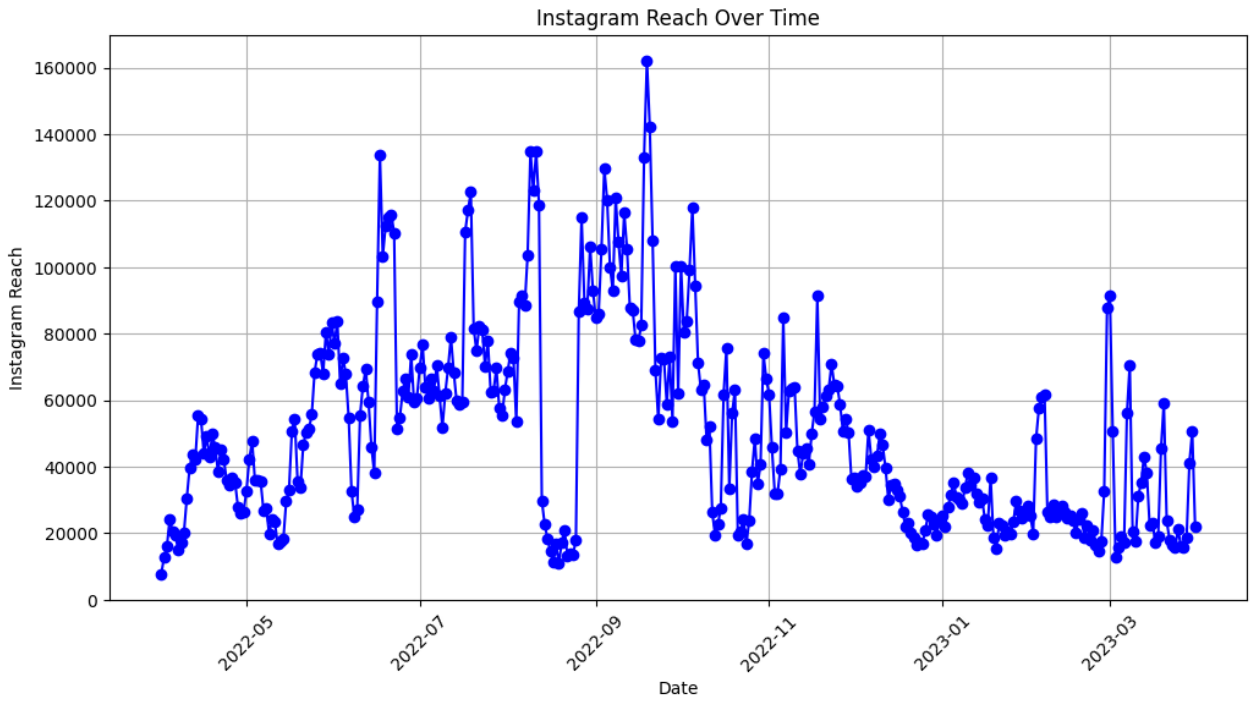
DATA VISUALISATION

```
mean_by_date = df.groupby(df['Date']).mean()
plt.figure(figsize=(12, 6))
plt.plot(mean_by_date.index,mean_by_date['Instagram reach'],label='Instagram Reach')
plt.xlabel('YEAR')
plt.ylabel('Instagram Reach Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



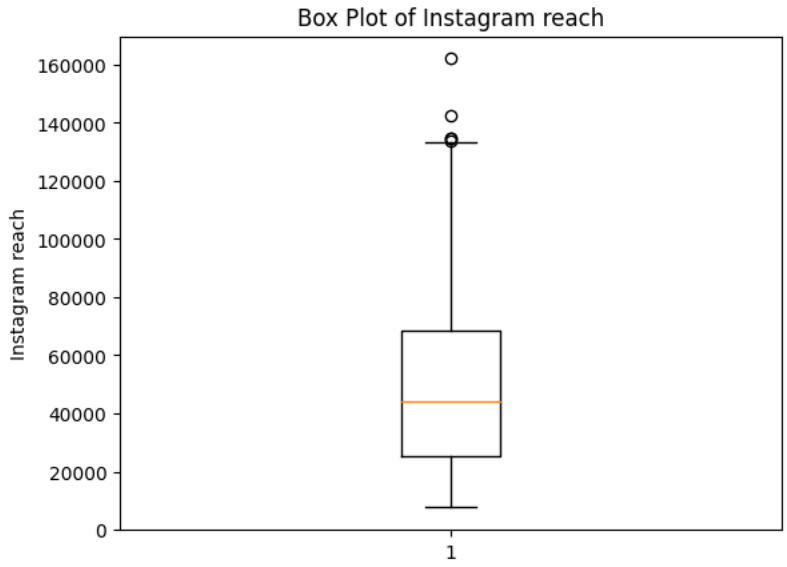

```
plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Instagram reach'], marker='o', linestyle='-', color='blue')
plt.xlabel('Date')
plt.ylabel('Instagram Reach')
plt.title('Instagram Reach Over Time')
```

```
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



BOX PLOT

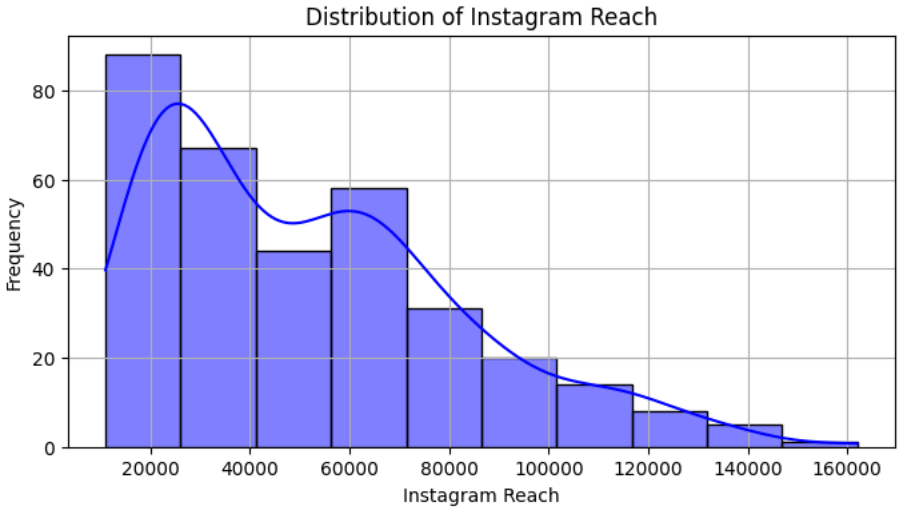
```
plt.boxplot(df['Instagram reach'])
plt.ylabel('Instagram reach')
plt.title('Box Plot of Instagram reach')
plt.show()
```



HISTOGRAM PLOT

```
plt.figure(figsize=(8, 4))
sns.histplot(df['Instagram reach'], bins=10, kde=True, color='blue')
```

```
plt.xlabel('Instagram Reach')
plt.ylabel('Frequency')
plt.title('Distribution of Instagram Reach')
plt.grid(True)
plt.show()
```

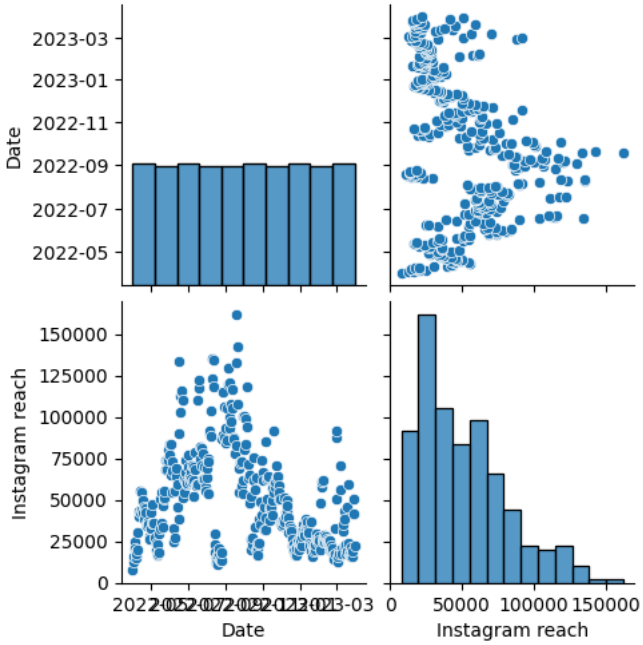


SCATTER PLOT

```
X = df['Date']
y = df['Instagram reach']

plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='steelblue', alpha=0.7)

plt.xlabel('Date')
plt.ylabel('Instagram Reach')
plt.title('Instagram Reach Over Time')
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.7)
plt.xticks(rotation=45)
plt.ylim(0, max(y) * 1.1)  # Adjusting the y-axis limits for better visibility
plt.show()
```

PAIR PLOT

```
sns.pairplot(df, vars=['Date','Instagram reach'])
plt.show()
```



GETTING THE ROLLING AVERAGE AS PER DIFFERENT TIME AND PERCENT CHANGE

```
df['Day'] = df['Date'].dt.day
df['Month'] = df['Date'].dt.month
df['Year'] = df['Date'].dt.year
df['Weekday'] = df['Date'].dt.weekday
df['Week_of_Year'] = df['Date'].dt.isocalendar().week

df['Rolling_Average_7Days'] = df['Instagram reach'].rolling(window=7).mean()
df['Rolling_Average_30Days'] = df['Instagram reach'].rolling(window=30).mean()

df['Percentage_Change'] = df['Instagram reach'].pct_change() * 100

df['Reach_Difference'] = df['Instagram reach'].diff()
df['Previous_Day_Reach'] = df['Instagram reach'].shift(1)
df = df.dropna()

print(df)
```

```
           Date  Instagram reach  Day  Month  Year  Weekday  Week_of_Year  \
29   2022-04-30            26410   30      4  2022        5            17
30   2022-05-01            32637    1      5  2022        6            17
31   2022-05-02            42204    2      5  2022        0            18
32   2022-05-03            47632    3      5  2022        1            18
33   2022-05-04            35793    4      5  2022        2            18
..          ...              ...  ...    ...   ...      ...           ...
360  2023-03-27            15622   27      3  2023        0            13
361  2023-03-28            18645   28      3  2023        1            13
362  2023-03-29            41238   29      3  2023        2            13
363  2023-03-30            50490   30      3  2023        3            13
364  2023-03-31            22014   31      3  2023        4            13

     Rolling_Average_7Days  Rolling_Average_30Days  Percentage_Change  \
29           31883.428571            33336.800000           0.863123
30           31392.142857            34170.700000          23.578190
31           32497.428571            35148.866667          29.313356
32           34048.857143            36203.000000          12.861340
```

```
33           34112.285714           36584.466667        -24.855139
..                    ...                    ...               ...
360          18085.571429           32338.066667         -2.848259
361          17368.857143           32368.700000         19.350915
362          20677.857143           32649.200000        121.174578
363          25559.428571           31410.566667         22.435618
364          26474.000000           29098.800000        -56.399287

     Reach_Difference  Previous_Day_Reach
29              226.0             26184.0
30             6227.0             26410.0
31             9567.0             32637.0
32             5428.0             42204.0
33           -11839.0             47632.0
..                ...                 ...
360            -458.0             16080.0
361            3023.0             15622.0
362           22593.0             18645.0
363            9252.0             41238.0
364          -28476.0             50490.0

[336 rows x 12 columns]
```

Rolling Average of Instagram Reach:

```
df['Rolling Average'] = df['Instagram reach'].rolling(window=7, min_periods=1).mean()
```

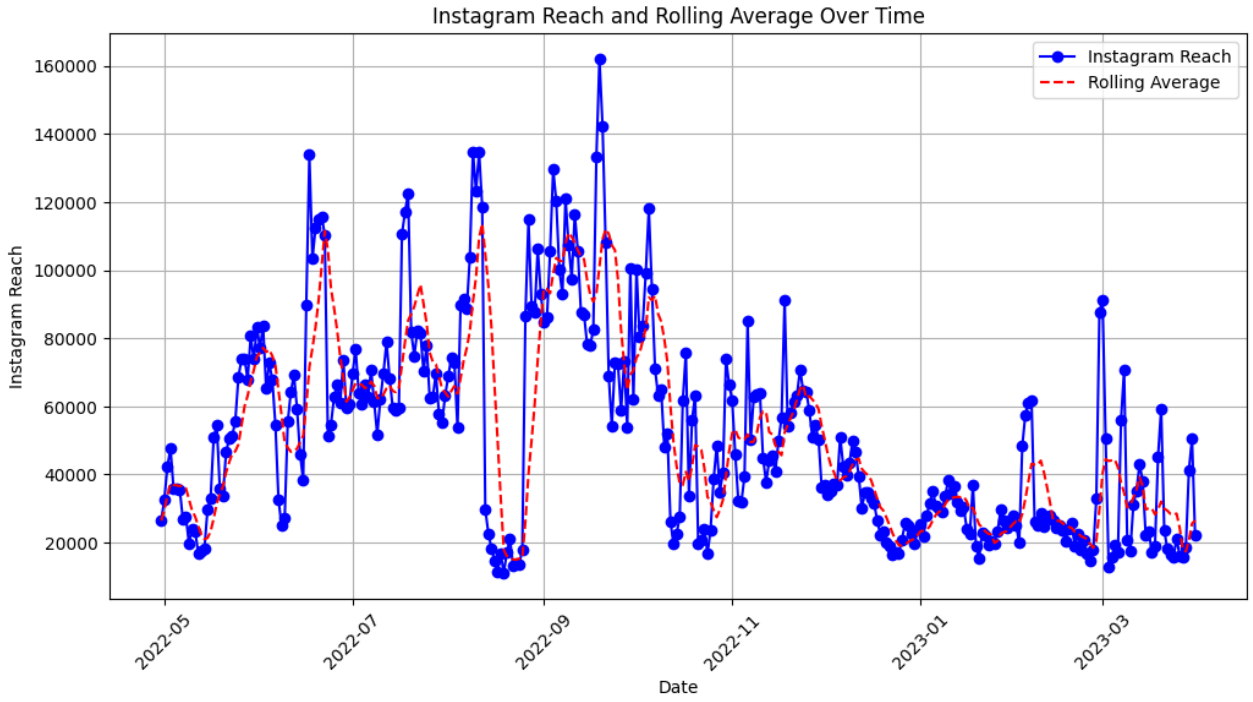
```
<ipython-input-100-597a4959adde>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df['Rolling Average'] = df['Instagram reach'].rolling(window=7, min_periods=1).mean()
```

```
df.head()
```

	Date	Instagram reach	Day	Month	Year	Weekday	Week_of_Year	Rolling_Average_7Days	Rolling_Average_30Days	Percentage_Cha
29	2022-04-30	26410	30	4	2022	5	17	31883.428571	33336.800000	0.863
30	2022-05-01	32637	1	5	2022	6	17	31392.142857	34170.700000	23.578
31	2022-05-02	42204	2	5	2022	0	18	32497.428571	35148.866667	29.313
32	2022-05-03	47632	3	5	2022	1	18	34048.857143	36203.000000	12.861
33	2022-05-04	35793	4	5	2022	2	18	34112.285714	36584.466667	-24.855

Line Plot of Instagram Reach and Rolling Average:

```
plt.figure(figsize=(12,6))
plt.plot(df['Date'], df['Instagram reach'], marker='o', linestyle='-', color='blue', label='Instagram Reach')
plt.plot(df['Date'], df['Rolling Average'], linestyle='--', color='red', label='Rolling Average')
plt.xlabel('Date')
plt.ylabel('Instagram Reach')
plt.title('Instagram Reach and Rolling Average Over Time')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```



ARIMA (AutoRegressive Integrated Moving Average)

```
pip install pmdarima
```

```
Requirement already satisfied: pmdarima in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.1)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.29.36)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.22.4)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.10.1)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.13.5)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.26.16)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2022.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.1.
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0
```

```
from pmdarima import auto_arima
stepwise_fit = auto_arima(df['Instagram reach'], trace=True,
suppress_warnings=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=7439.869, Time=0.34 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=7464.605, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=7466.566, Time=0.06 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=7466.560, Time=0.07 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=7462.607, Time=0.02 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=7437.179, Time=0.70 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=7454.999, Time=0.20 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=7442.863, Time=0.48 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=7439.400, Time=0.39 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=7448.128, Time=0.14 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=7439.422, Time=0.26 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=7440.297, Time=1.07 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=7435.216, Time=0.22 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=7452.992, Time=0.12 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=7440.946, Time=0.16 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=7437.153, Time=0.31 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=7437.432, Time=0.33 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=7464.556, Time=0.06 sec
 ARIMA(0,1,3)(0,0,0)[0]             : AIC=7446.128, Time=0.13 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=7436.065, Time=0.21 sec
```

```
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=7438.319, Time=0.69 sec

Best model:  ARIMA(1,1,2)(0,0,0)[0]
Total fit time: 6.043 seconds
```

SPLITTING THE DATASET

```
train_data = df.iloc[:-10] # selecting all rows from the DataFrame
test_data = df.iloc[-10:] # selecting the last 10 rows of the DataFrame
```

TRAINING OUR MODEL

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(train_data['Instagram reach'], order=(1, 1, 2))
model = model.fit()
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: An unsupported index was provi
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: An unsupported index was provi
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: An unsupported index was provi
  self._init_dates(dates, freq)
```

SARIMAX Results

Dep. Variable:	Instagram reach	**No. Observations:**	326
Model:	ARIMA(1, 1, 2)	**Log Likelihood**	-3605.344
Date:	Mon, 17 Jul 2023	**AIC**	7218.687
Time:	15:18:33	**BIC**	7233.822
Sample:	0	**HQIC**	7224.728
	- 326		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.6814	0.069	9.924	0.000	0.547	0.816
ma.L1	-0.7528	0.074	-10.137	0.000	-0.898	-0.607
ma.L2	-0.1859	0.056	-3.344	0.001	-0.295	-0.077
sigma2	2.659e+08	2.15e-10	1.24e+18	0.000	2.66e+08	2.66e+08

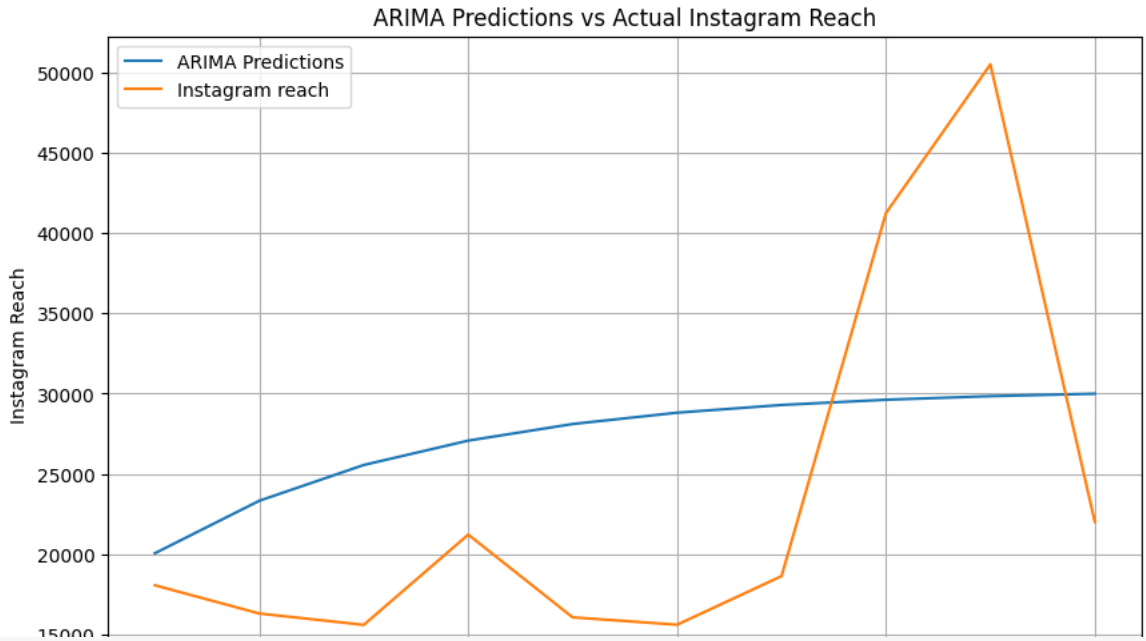
Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	177.27
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.48	**Skew:**	0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.60

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.89e+33. Standard errors may be unstable.

GRAPH PLOTTING OF ACTUAL VS PREDICTION

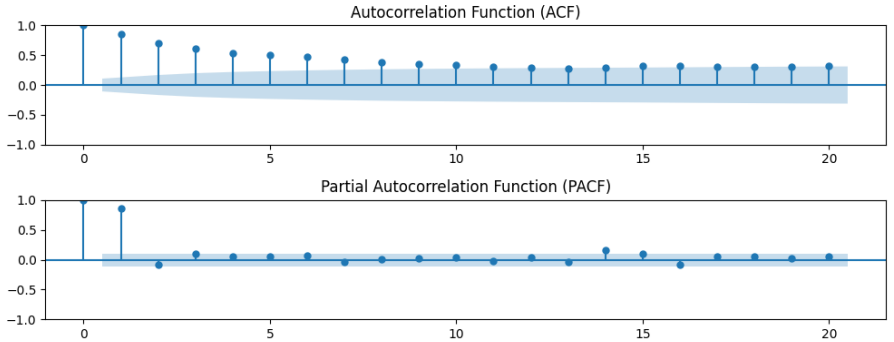
```
start = len(train_data)
end = len(train_data) + len(test_data) - 1
pred = model.predict(start=start, end=end, typ='levels').rename('ARIMA Predictions')
pred.index = test_data.index  # Aligning the index of predicted values with test_data
pred.plot(legend=True,figsize=(10, 6))
test_data['Instagram reach'].plot(legend=True)
plt.xlabel('Date')
plt.ylabel('Instagram Reach')
plt.title('ARIMA Predictions vs Actual Instagram Reach')
plt.grid(True)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:834: ValueWarning: No supported index is available. Prediction
  return get_prediction_index(
```




```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

data = df['Instagram reach']

fig, axes = plt.subplots(2, 1, figsize=(10, 4))

# ACF plot
plot_acf(data, lags=20, ax=axes[0])
axes[0].set_title('Autocorrelation Function (ACF)')

# PACF plot
plot_pacf(data, lags=20, ax=axes[1])
axes[1].set_title('Partial Autocorrelation Function (PACF)')

plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/graphics/tsaplots.py:348: FutureWarr
  warnings.warn(
```

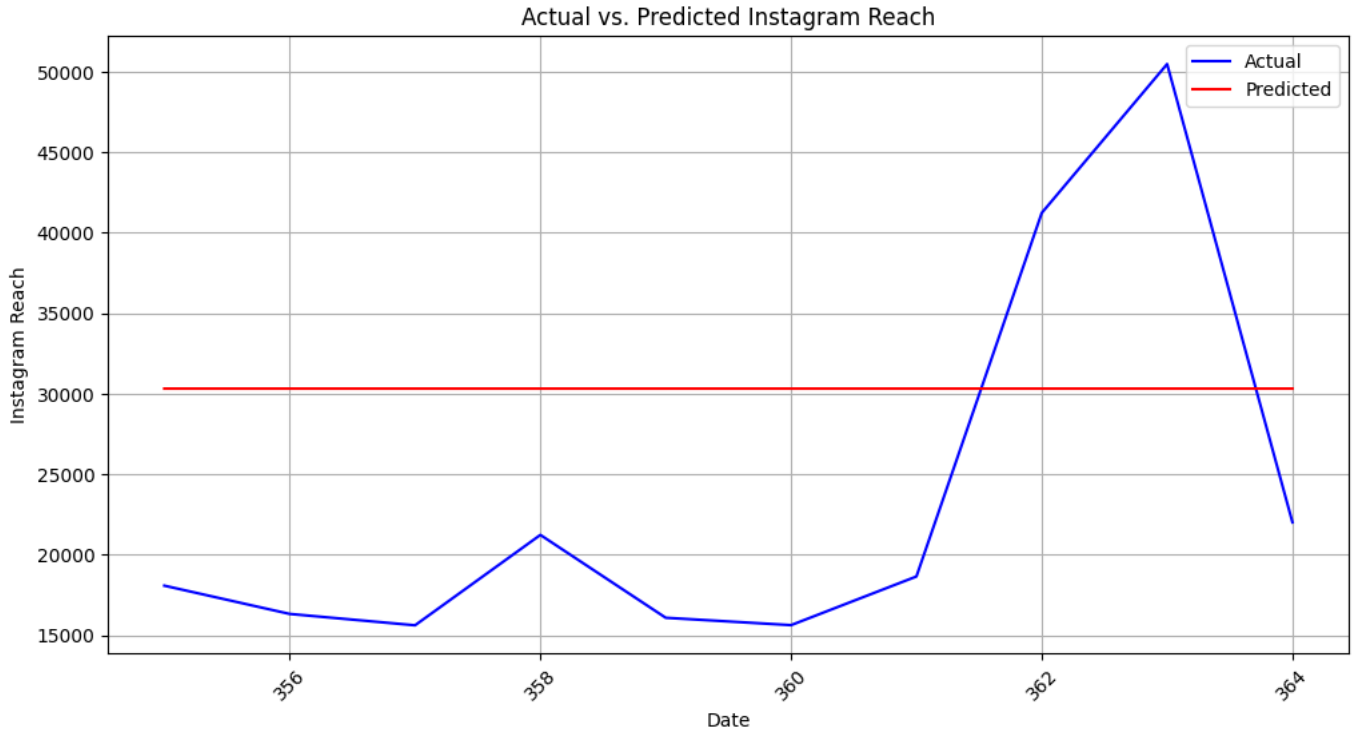



```
predictions = model.predict(start=test_data.index[0], end=test_data.index[-1])

mse = mean_squared_error(test_data['Instagram reach'], predictions)
mae = mean_absolute_error(test_data['Instagram reach'], predictions)
print(mse)
print(mae)
```

```
179512869.0885865
13005.44191363524
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:834: ValueWarning: No supported index is available. Predictior
  return get_prediction_index(
```

Plot the actual vs. predicted Instagram reach

```
plt.figure(figsize=(12, 6))

# Plotting the actual Instagram reach
plt.plot(test_data.index, test_data['Instagram reach'], color='blue', label='Actual')

# Plotting the predicted Instagram reach
plt.plot(predictions.index, predictions, color='red', label='Predicted')
plt.xlabel('Date')
plt.ylabel('Instagram Reach')
plt.title('Actual vs. Predicted Instagram Reach')
plt.legend()
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



✓ 1s completed at 9:19 PM