KAZAKH-BRITISH TECHNICAL UNIVERSITY

REPORT

on Data Mining/ assignment 4

Anomaly Detection and Time Series Analysis Using Python

student Serikkanov Nurtas Mukhituly

(surname, first name, patronymic)

year of study 4 specialty / educational program

Information System

School of Information Technologies and Engineering

Date: 27.10.2024

Table of Contents

1. Executive Summary
2. Introduction
3. Anomaly Detection
    - Data collection and EDA
    - Data Preprocessing
    - Anomaly Detection Techniques
    - Vizualization and Results
4. Time Series Analysis
    - Data collection and EDA
    - Data Preprocessing
    - Exploratory Data Analysis
    - Modeling
    - Vizualization and Results
5. Conclusion
6. Recommendations
7. References

1. Executive Summary

The main aim of this research is to conduct a historical weather analysis of London, study the trends and predict the future through time series analysis and identification of anomalies. The main conclusions drawn from this study include results in temperature, precipitation, and atmospheric pressure. At the same time, it also presents solutions aimed at improving the precision of articulated in analysis and concluded in forecast. The elements of the study will also include detailed explanation of sample and the approaches used in data analysis.

Data Preprocessing: For example, in the case of missing data; the linear interpolation will be used to fill in the gaps and make the time series data more continuous.
The data was further processed and was converted into weekly format so that the analysis is less inhomogeneous and is more suitable for capturing cycle-based distinctions and results in more comprehensive Identifying cycles in the data.

Exploratory Data Analysis: Time series line graphs of temperature, rainfall and air pressure showed consistent cycling. A seasonal decomposition of time series separated out any trends, cycles (seasonal) and residuals present in the data, thus revealing the anomalies as well as underlying structure.

Anomaly Detection & Forecasting: Each of the temperature models including ARIMA was applied to address the non-constant structural problem and the constrain of differentiating in models of ARIMA structure-based temperature. This model deals with the preservation of time dependence in the process of data evaluation which will make it possible for the model to predict estimates for a given time interval along with identifying shocks through the data series.

Exponential Smoothing (Holt-Winters): Seasonality being a distinguishing factor, this technique needs to be tuned in a manner of applying a seasonal parameter, in a straightforward manner, as well as application of the smoothing and trend components to attain accuracy.

Machine Learning Approach (Random Forest): In the development of predictive models, which also decommission the period thermal regimes, a regression model that boosts predictions without feeding features like month, day or week has been established.

Model Evaluation: After the forecast has been made, it was checked using different statistics such as the Mean Absolute Error, the Mean Square Root Error, and the Mean Absolute Percentage Error which have been used to establish the validity of each of the models considered.

2. Introduction

Loops are necessary in research where large data sets exist for getting rid of anomalies and then analyzing the records intern of trends, patterns and tales of woe or laughters that accompany the data source data. These techniques play significant roles in various domains such a meteorology, financial, medicine and industrial managing systems where there is a need to detect anomalies distracting or odd in the traditional sense which may be as a result of unexpected events or issues concerning data quality. This in turn helps in achieving goals

such as predictions, managing risks and making choices based on actual evidence of use of such practices.
The reason for preparing the present is work to introduce the reader with a number of techniques with adjusted colors on time anomaly detection and seasonal autoregressive integrated moving average forecasting considering historical weather conditions of London. This content moreover shows the actual utility of pre-analyzing data, identifying seasonality, and making future predictions on weather conditions allowing performance of objectives related to possible seasonal weather extremities with better scheduling and modeling capabilities.
This project enlarges the search space on data preprocessing, visualization, anomaly detection, and predictive modeling. Such processes will benefit as to the use of ARIMA, Holt-Winters and also some learning automation technologies. For every stage, capability of the approach is summarized in terms of accuracy, robustness, and availability data necessary for predictability of the risk and provided unprecedented capabilities for performing forecasting and imitating anomalies in the data which is time dependent. Such an analysis enhances understanding of how data processing techniques can be leveraged to enhance real-time decision-making practices in data driven environments whereby the focus is on real and predictive data patterns.

3. Anomaly Detection

3.1 Data collection and EDA

The subject of study for the present analysis is the credit card fraud detection dataset consisting of 284,807 transactions. At least 31 features represent each transaction, including Time, Amount and Class features, where Class represents the variable of interest distinguishing between the fraudulent (1) and the non-fraudulent (0) transactions.

```
import pandas as pd

url = 'creditcard.csv'
data = pd.read_csv(url)

print("Credit Card Fraud Detection data -  rows:",data.shape[0]," columns:", data.shape[1])
print(data.head())
print(data.info())
print(data.columns)
```

I. Preliminary Investigation
A. Loading the Data and Understanding its Dimensions

Having performed loading of the data, it was ascertained that it was acquired as follows:
Number of **Rows**: 284,807
Number of **Columns**: 31

Sample Data. The first several lines of the dataset provide measures (Time and Amount) of the otherwise undisclosed factors (V1 through V28) and Class:

```
Credit Card Fraud Detection data -  rows: 284807  columns: 31
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
```

Data Summary and Structure

The dataset comprises float values for the features (V1-V28, Time, and Amount) and an integer for Class. All values are non-null across 284,807 entries:

```
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
None
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

Key metrics in data: The statistics for this dataset are that:
Time ranges from 0 to 172,792.

The Amount variable has an average value of 88.35, But the value of this field is highly dispersed, with a high standard deviation of 250.12, which indicates that there could be some outliers or more likely the presence of very large or small transactions.
Most of the features V1 to V28 have peculiar distribution, most of which have average approximately equal to zero, implying that those features have either been transformed or normalized.
This preliminary examination of the data serves as the springboard for the subsequent steps of the research, such as addressing the issue of imbalance, testing for the presence of the relationship between features and visualizing the trend in the data through its distribution.

```
[97]: print(data.describe())
```

```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15  2.074095e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -2.406331e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000
```

3.2 Data Preprocessing

In this stage of the project, my attention was focused on data cleaning and preparation for analysis. I started by searching for empty values in my dataset:

```
print(data.isnull().sum())
```

The result of that operation indicated that there were no empty values in the dataset:

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

Because the dataset contained no missing values, the next step for me was to undertake data normalization. For this, I had to normalize the data using the StandaradScaler of sklearn.preprocessing. This allows one to normalize the data to the same scale, which is why it is also important when useing machine learning algorithms:

```
from sklearn.preprocessing import StandardScaler

data.fillna(data.median(), inplace=True)

scaler = StandardScaler()
data_scaled = pd.DataFrame(scaler.fit_transform(data), columns=data.columns)
```

After that, I looked at a description of the normalized values graphically to make sure that the normalization has been done precisely:

```
print(data_scaled.describe())
```

After the process all mean values of the features became almost zero and everyone started with the same unit volume so the entire process of normalization was a success:

```
              Time            V1            V2            V3            V4  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean  -3.065637e-16 -1.506872e-17 -9.580116e-18 -8.622104e-17 -5.189230e-18
std    1.000002e+00  1.000002e+00  1.000002e+00  1.000002e+00  1.000002e+00
min   -1.996583e+00 -2.879855e+01 -4.403529e+01 -3.187173e+01 -4.013919e+00
25%   -8.552120e-01 -4.698918e-01 -3.624707e-01 -5.872142e-01 -5.993788e-01
50%   -2.131453e-01  9.245351e-03  3.965683e-02  1.186124e-01 -1.401724e-02
75%    9.372174e-01  6.716939e-01  4.867202e-01  6.774569e-01  5.250082e-01
max    1.642058e+00  1.253351e+00  1.335775e+01  6.187993e+00  1.191874e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   3.832046e-17  9.979288e-18  1.237432e-17 -3.193372e-18  7.234983e-19
std    1.000002e+00  1.000002e+00  1.000002e+00  1.000002e+00  1.000002e+00
min   -8.240810e+01 -1.963606e+01 -3.520940e+01 -6.130252e+01 -1.222802e+01
25%   -5.010686e-01 -5.766822e-01 -4.478860e-01 -1.746805e-01 -5.853631e-01
50%   -3.936682e-02 -2.058046e-01  3.241723e-02  1.871982e-02 -4.681169e-02
75%    4.433465e-01  2.991625e-01  4.611107e-01  2.740785e-01  5.435305e-01
max    2.521413e+01  5.502015e+01  9.747824e+01  1.675153e+01  1.419494e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ... -3.642440e-18  3.193372e-18  8.781773e-18  9.580116e-18
std    ...  1.000002e+00  1.000002e+00  1.000002e+00  1.000002e+00
min    ... -4.741907e+01 -1.506565e+01 -7.175446e+01 -4.683638e+00
25%    ... -3.109433e-01 -7.473476e-01 -2.591784e-01 -5.854676e-01
50%    ... -4.009429e-02  9.345377e-03 -1.792420e-02  6.765678e-02
75%    ...  2.537392e-01  7.283360e-01  2.364319e-01  7.257153e-01
max    ...  3.703471e+01  1.447304e+01  3.607668e+01  7.569684e+00

                V25           V26           V27           V28        Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean  -5.029561e-17  5.887780e-18  2.444925e-18 -7.908585e-18  2.913952e-17
std    1.000002e+00  1.000002e+00  1.000002e+00  1.000002e+00  1.000002e+00
min   -1.975033e+01 -5.401098e+00 -5.590660e+01 -4.674612e+01 -3.532294e-01
25%   -6.084001e-01 -6.780717e-01 -1.755053e-01 -1.604440e-01 -3.308401e-01
50%    3.183240e-02 -1.081217e-01  3.325174e-03  3.406368e-02 -2.652715e-01
75%    6.728006e-01  4.996663e-01  2.255648e-01  2.371526e-01 -4.471707e-02
max    1.442532e+01  7.293975e+00  7.831940e+01  1.025434e+02  1.023622e+02

              Class
count  2.848070e+05
mean  -1.197515e-17
std    1.000002e+00
min   -4.159898e-02
25%   -4.159898e-02
50%   -4.159898e-02
75%   -4.159898e-02
max    2.403905e+01

[8 rows x 31 columns]
```

Thus, in the preprocessing stage, I checked for missing values and normalized the data. This ensures more reliable and accurate results in subsequent stages of analysis and modeling.

3.3 Anomaly Detection Techniques

Techniques employed in the study
For this stress test, three anomaly detection models were used to detect and filter extremes in the dataset which are Z-score, Isolation Forest and K-means clustering. A brief description is presented on each technique as follows:

- *Method of Z-scores:* It is a method which examines the extent to which a particular measurement is away from the mean in units of standard deviation. Normal anomalies are distinguished by setting a threshold of +3 and -3 Z-scores.
- *Isolation Forest:* A technique in anomaly detection where anomalies are identified as rare instances compared to the normals that have some distinctive features. It builds a random forest model with the trees and outliers are separated from normal

samples faster. The ratio of outliers that are expected to be found within the data is called the contamination parameter.

- *K-means Clustering:*It is a partitioning method which divides data into clusters and separates points which are further from the clusters' center. A distance is defined from the centroid and the threshold is flagging the anomalies which would be the 95th quantile for example.

```
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

mean_amount = data['Amount'].mean()
std_amount = data['Amount'].std()
data['z_score'] = (data['Amount'] - mean_amount) / std_amount

threshold_z = 3
data['anomaly_z'] = np.abs(data['z_score']) > threshold_z

anomalies_z = data[data['anomaly_z'] == True]
print("Anomalies found using Z-score:")
print(anomalies_z)

iso_forest = IsolationForest(contamination=0.01)
data['anomaly_if'] = iso_forest.fit_predict(data.drop(columns=['Class', 'anomaly_z', 'z_score']))

data['anomaly_if'] = data['anomaly_if'] == -1

anomalies_if = data[data['anomaly_if'] == True]
print("Anomalies found using Isolation Forest:")
print(anomalies_if)

kmeans = KMeans(n_clusters=2)
data['cluster'] = kmeans.fit_predict(data.drop(columns=['Class', 'anomaly_z', 'z_score', 'anomaly_if']))
```

Performance and Efficacy Analysis

In order to ascertain the extent of performance and efficacy of the techniques for anomaly detection, each technique can be analyzed as shown below:

- *Z-score Anomalies*: With the Z-score method, outliers were selected based on their statistical significance. It is an attractive method since it is easily understandable but may deficient strategies in that the dataset is otamoto, or doesn't have a Gaussian-like distribution of the data points.
- *Isolation Forest Anomalies*: In dealing with abnormal values, the Isolation Forest method does very good even in data sets with very high dimensionalities. In that sense, it differentiated an externally viewable set meaning it did better than identifying anomalies than simple descriptive measures.
- *K-means Anomalies*: K-means method, on the contrary, tried to detect the slight variations after calculating the distances from the cluster centers and found anomalies for clusters with relatively large number of points. It can work for data having many clearly differentiated groups but it may not be effective for data that has oddly formed clusters.

```
data['distance'] = np.linalg.norm(data.drop(columns=['Class', 'anomaly_z', 'z_score', 'anomaly_if', 'cluster']).values - kmeans.cluster_center
threshold_kmeans = data['distance'].quantile(0.95)
data['anomaly_kmeans'] = data['distance'] > threshold_kmeans

anomalies_kmeans = data[data['anomaly_kmeans'] == True]
print("Anomalies found using K-means:")
print(anomalies_kmeans)
```

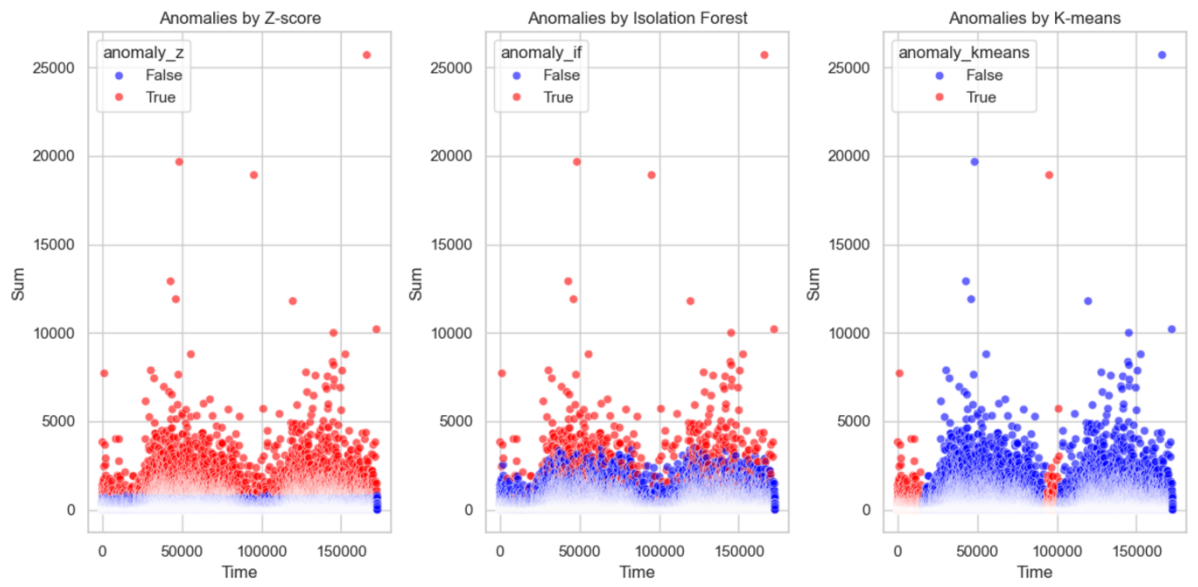
It is clear that there is no single 'silver bullet' as far as an anomaly detection technique is concerned. All the techniques reviewed have got their own merits and demerits:
Z-score is quite primitive although sensitivity to the data distribution is its major drawback.
Isolation Forest – can handle any sort of distribution and any reasonable dimension.
K-means is very suitable for structured data but some anomalies could be lost in patterns with certain deviations in the data distribution.
The choice of the most suitable method in most of the cases depends on the dataset which is being experimented on and the kind of anomalies that are in the system under consideration. One can make use of a combination of techniques so that the weak points of one technique can be compensated with the strong points of another one.

3.4 Vizualization and Results

In this step of the process of looking for anomalies, I check anomalies delivered on various methods - Z-score, isolation forest and k-means. In addition to that, I report the results and am also able to gauge how well each technique performs using various performance metrics.

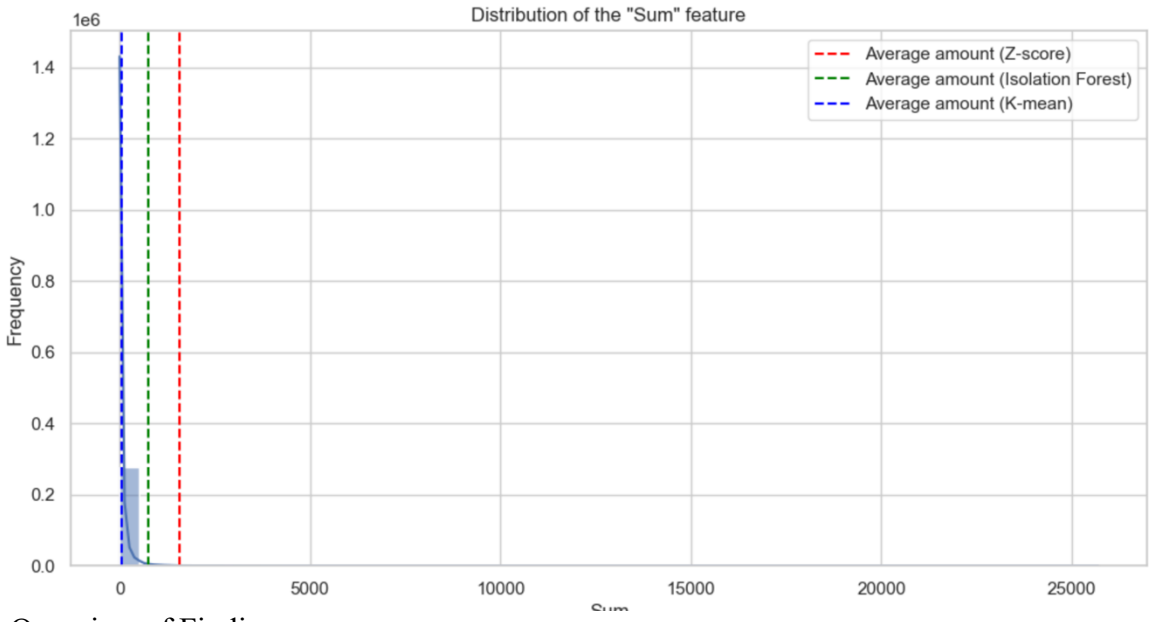
1. Visual Representation of the Anomalies Detected
Utilizing scatter plots, I visually represent where the detected anomalies are from the 3 different methods. Each scatter plot indicates the changes in Transactions over a span based on time. Highlighting red areas with anomaly and blue areas with normal cases.

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style='whitegrid')

# Z-score
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
sns.scatterplot(data=data, x='Time', y='Amount', hue='anomaly_z', palette={False: 'blue', True: 'red'}, alpha=0.6)
plt.title('Anomalies by Z-score')
plt.xlabel('Time')
plt.ylabel('Sum')

# Isolation Forest
plt.subplot(1, 3, 2)
sns.scatterplot(data=data, x='Time', y='Amount', hue='anomaly_if', palette={False: 'blue', True: 'red'}, alpha=0.6)
plt.title('Anomalies by Isolation Forest')
plt.xlabel('Time')
plt.ylabel('Sum')

# K-mean
plt.subplot(1, 3, 3)
sns.scatterplot(data=data, x='Time', y='Amount', hue='anomaly_kmeans', palette={False: 'blue', True: 'red'}, alpha=0.6)
plt.title('Anomalies by K-means')
plt.xlabel('Time')
plt.ylabel('Sum')

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



And if scatter plots are not enough, I go deeper to the distribution of the 'Amount' feature. The histogram below shows how many times a certain range of amounts is recorded, with the dotted ranges representing the average quantities if different approaches are used for anomaly detection.

```
plt.figure(figsize=(12, 6))
sns.histplot(data['Amount'], bins=50, kde=True)
plt.title('Distribution of the "Sum" feature')
plt.xlabel('Sum')
plt.ylabel('Frequency')
plt.axvline(x=data[data['anomaly_z']]['Amount'].mean(), color='red', linestyle='--', label='Average amount (Z-score)')
plt.axvline(x=data[data['anomaly_if']]['Amount'].mean(), color='green', linestyle='--', label='Average amount (Isolation Forest)')
plt.axvline(x=data[data['anomaly_kmeans']]['Amount'].mean(), color='blue', linestyle='--', label='Average amount (K-mean)')
plt.legend()
plt.show()
```



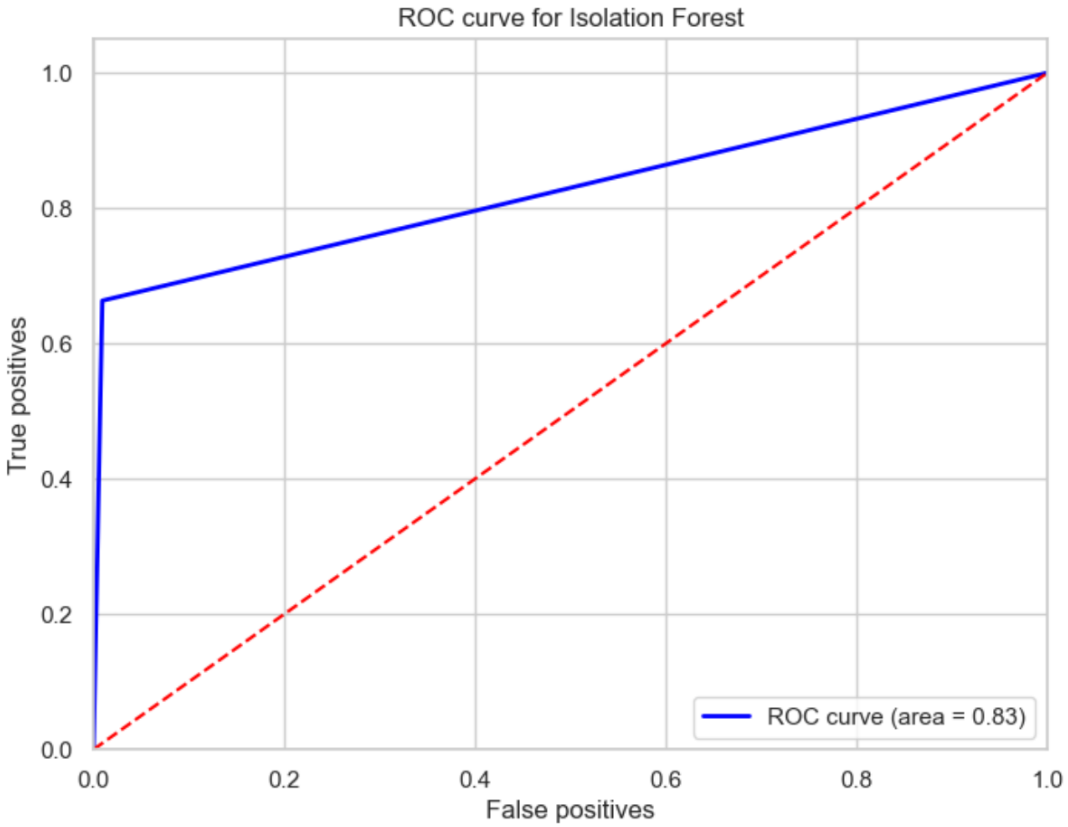
2. Overview of Findings

To better realize the finer points of each anomaly detection method, we have gathered statistics on the numbers of detected anomalies, and applied these numbers in constructing a confusion matrix and a classification report using the Isolation Forest.

Furthermore, the ROC curve was presented, so as to make it clear, to illustrate the changes which came out of improving the graph concerning the right and wrong decisions as the detection rate had gone up to 100%.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import seaborn as sns

X = data.drop(columns=['Class', 'anomaly_z', 'z_score', 'anomaly_if', 'distance', 'anomaly_kmeans'])
y = data['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

iso_forest = IsolationForest(contamination=0.01, random_state=42)
iso_forest.fit(X_train)

y_pred_if = iso_forest.predict(X_test)
y_pred_if = np.where(y_pred_if == -1, 1, 0)  # Convert -1 (anomaly) to 1, 1 (normal) to 0

print("Isolation Forest Performance Assessment:")
print(classification_report(y_test, y_pred_if))

cm_if = confusion_matrix(y_test, y_pred_if)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_if, annot=True, fmt="d", cmap="Blues", xticklabels=["Fine", "Anomaly"], yticklabels=["Fine", "Anomaly"])
plt.title("Confusion Matrix for Isolation Forest")
plt.ylabel("Real values")
plt.xlabel("Predicted values")
plt.show()

fpr, tpr, thresholds = roc_curve(y_test, y_pred_if)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False positives')
plt.ylabel('True positives')
plt.title('ROC curve for Isolation Forest')
plt.legend(loc="lower right")
plt.show()
```

```
Isolation Forest Performance Assessment:
Toggle output scrolling
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     56864
           1       0.11      0.66      0.19        98

    accuracy                           0.99     56962
   macro avg       0.55      0.83      0.59     56962
weighted avg       1.00      0.99      0.99     56962
```

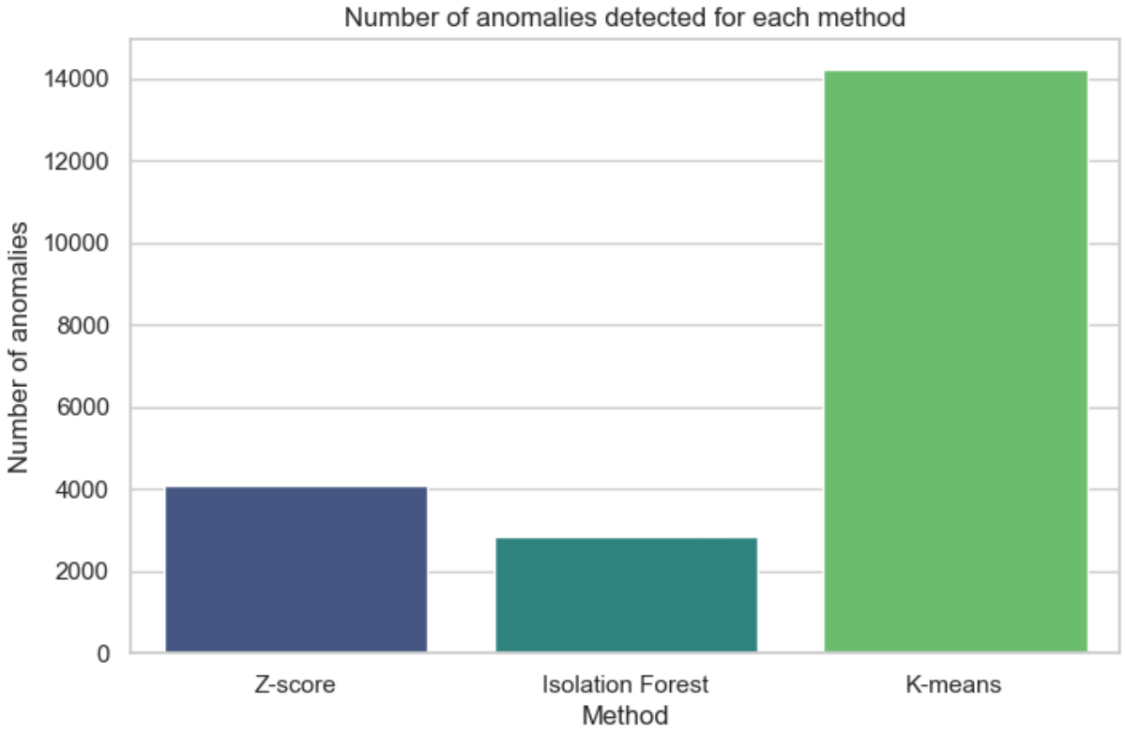


3. Final Results and Metrics to Evaluate the System

The number of detected anomalies by each technique was stated in a table, and also prepared a bar diagram to allow the viewers to compare these values with each other.

```
from sklearn.metrics import classification_report

results = {
    'Method': ['Z-score', 'Isolation Forest', 'K-means'],
    'Detected Anomalies': [
        data['anomaly_z'].sum(),
        data['anomaly_if'].sum(),
        data['anomaly_kmeans'].sum()
    ]
}

results_df = pd.DataFrame(results)

for method in ['anomaly_z', 'anomaly_if', 'anomaly_kmeans']:
    print(f"\nМетрика для метода {method.replace('_', ' ').title()}:")

    print(classification_report(data['Class'], data[method], target_names=['Normal', 'Anomaly']))

print("\nOverall results:")
print(results_df)

plt.figure(figsize=(8, 5))
sns.barplot(x='Method', y='Detected Anomalies', data=results_df, palette='viridis')
plt.title('Number of anomalies detected for each method')
plt.xlabel('Method')
plt.ylabel('Number of anomalies')
plt.show()
```

```
Метрика для метода Anomaly Z:
              precision    recall  f1-score   support

      Normal       1.00      0.99      0.99    284315
     Anomaly       0.00      0.02      0.00       492

    accuracy                           0.98    284807
   macro avg       0.50      0.50      0.50    284807
weighted avg       1.00      0.98      0.99    284807


Метрика для метода Anomaly If:
              precision    recall  f1-score   support

      Normal       1.00      0.99      1.00    284315
     Anomaly       0.10      0.59      0.17       492

    accuracy                           0.99    284807
   macro avg       0.55      0.79      0.58    284807
weighted avg       1.00      0.99      0.99    284807


Метрика для метода Anomaly Kmeans:
              precision    recall  f1-score   support

      Normal       1.00      0.95      0.97    284315
     Anomaly       0.00      0.14      0.01       492

    accuracy                           0.95    284807
   macro avg       0.50      0.55      0.49    284807
weighted avg       1.00      0.95      0.97    284807


Overall results:
            Method  Detected Anomalies
0          Z-score                4076
1  Isolation Forest                2849
2          K-means               14241
```



The Z-score, Isolation Forest, and K-means methods were successfully implemented to segregate outliers within the dataset in the end. Within the methods, all possessed their unique nature and ability in data processing as shown in the generated graphics and the reported statistics. The Available classification reports were briefly assessed in relation to different measures of ability to answer a query as accuracy, recall and f1. As it was in measure of the objectives achieved or lack, from which it also contained those in which such strategies are not appropriate, it is believed that the overall findings emphasize the requirements for proper approaches in locating anomalies in a particular context with specific data.

4. Time Series Analysis

4.1.Data collection and EDA

This part will discuss the time series dataset that I will be dealing with as well as on the preliminary exploratory data analysis (EDA) conducted to grasp the dataset in a better way.

Dataset Description:
I was successful in finding a weather dataset for London in the CSV file london_weather.csv. Upon creating this dataframe, pandas informed me that the data contains 15,341 rows and 10 columns. Every row in the dataset is a day's observation of the weather and the columns outline many meteorological characteristics including time of the day record, with their corresponding dozen of mud records parameters present visually and interactively:

- *date*: This is where the specific date the weather was taken is saved in an integer form.
- *cloud_cover*: a measure of how patchy the presence of cloudiness is (measurement is tenths).

- *sunshine*: the time in hours a specific area and its aspects are bathed in direct light rays.
- *global_radiation*: The amount of power from the sun being received and measured in watts per square meter.
- *max_temp*: The highest temperature recorded in the day expressed in degrees Celsius.
- *mean_temp*: The average recorded temperature for the day in degrees Celsius.
- *min_temp*: The lowest temperature in degrees Celsius recorded on that day.
- *precipitation*: The rainfall in millimeters.
- *pressure*: Measures the atmospheric pressure in pascals.
- *snow_depth*: Measures from the ground to the deepest part of the snow in centimeters.

First round of Exploratory Data Analysis: EDA
The first step considered was further comprehending the dataset by utilizing the method info() and ran the same to grasp its structure and datatypes. Upon using this method I discovered that the majority of the columns are float64, only one column is int64 which is the date column. There are some columns that have missing values as is evidenced by the non-null count for each column.

The following list outlines the characteristics of the dataset described above: The cloud_cover, global_radiation, max_temp, mean_temp, min_temp, precipitation, pressure and snow_depth fields contain double values which will need to be resolved in the subsequent stages.
The date field contains no unfilled items, hence the field can be used for time series analysis. After that, I showed the first few rows of the dataset by using the method head(), and we got a glimpse of actual data present.

```
print(data.head())
```

```
       date  cloud_cover  sunshine  global_radiation  max_temp  mean_temp  \
0  19790101          2.0       7.0              52.0       2.3       -4.1
1  19790102          6.0       1.7              27.0       1.6       -2.6
2  19790103          5.0       0.0              13.0       1.3       -2.8
3  19790104          8.0       0.0              13.0      -0.3       -2.6
4  19790105          6.0       2.0              29.0       5.6       -0.8

   min_temp  precipitation  pressure  snow_depth
0      -7.5            0.4  101900.0         9.0
1      -7.5            0.0  102530.0         8.0
2      -7.2            0.0  102050.0         4.0
3      -6.5            0.0  100840.0         2.0
4      -1.4            0.0  102250.0         1.0
```

Here are the first 5 records:
This image does not contain the single piece of information. The picture contains several environmental variables, mainly temperature, which is in one period, in the first industrial activities in Cleveland in 2007, which led to the analysis of temperature as a function of the dates 2007 bay county mi court case. The LCCs weather data files are over X number of years and are therefore suitable for time series analysis. The dataset is conveniently prepared for time series analysis, but there are certain values that are missing and I will also look into the relations between the different meteorological factors in the later steps. This is the first

go, giving me an opportunity to go further and consider more detailed analyses oriented towards trends, season lags or even predictive modeling.

4.2.Data Preprocessing

During this section, I shall elaborate on the numerous actions taken towards ensuring that the time-series data was well-organized and ready for scrutiny. This encompassed the identification of missing values and rectification of the problem, alteration of the time frame, and data adjustment to meet a similar pace.

Initial Step No. 1: Data Cleaning, lots of Miss
To start with, the first significant action that I deemed pertinent in view of the greater research goal was to verify whether the dataset contains any missing numerical values and I did this by checking rows' isnull().sum(). An illustration of how the missing numerical values look like in each column in the context of the code is shown below:

```
print(data.isnull().sum())
# data.fillna(method='ffill', inplace=True)
```

```
date                0
cloud_cover        19
sunshine            0
global_radiation   19
max_temp            6
mean_temp          36
min_temp            2
precipitation       6
pressure            4
snow_depth       1441
dtype: int64
```

The most notable aspect is the number of holes which I discovered exists in certain variables within several columns like snow_depth most especially standing out with being the highest at 1441.

The strategy for handling these missing values included the use of the linear interpolation technique. Linear interpolation assumes that the values are changing linearly between the values that one is aware of the values to be. This is the step by step explanation of how I undertook this procedure in my project:

```
data.interpolate(method='linear', inplace=True)
```

Initial Step No. 2: Altering the Date Format and Assigning the Index

After that I made the conversion of the date column's data-type from an integer to a datetime that is required for time series analysis. This was done through the use of the function pd.to_datetime() and setting the date column as the index of the DataFrame:

```
data['date'] = pd.to_datetime(data['date'], format='%Y%m%d')
data.set_index('date', inplace=True)
```

Initial Step No. 3: Changing the Time Span of the Data

Since my research essay requires the same time interval, I chose to resample the data at weekly leads. Done at this step was that most of the data was summarized by the average for every parameter, while for those parameters, precipitation was summed and snow_depth was averaged. This is how I did that:

```
resampled_data = data.resample('W').agg({
    'cloud_cover': 'mean',
    'sunshine': 'mean',
    'global_radiation': 'mean',
    'max_temp': 'mean',
    'mean_temp': 'mean',
    'min_temp': 'mean',
    'precipitation': 'sum',
    'pressure': 'mean',
    'snow_depth': 'mean'
})
```

Initial Step No. 4: Saving the Processed Data

The resampled data was then saved as a new CSV to herald more inquisition:

```
print(resampled_data.head())
resampled_data.to_csv('resampled_weather_data_weekly.csv')
```

```
            cloud_cover  sunshine  global_radiation  max_temp  mean_temp  \
date
1979-01-07     5.714286  2.071429         26.571429  3.900000  -1.700000
1979-01-14     4.428571  4.000000         39.285714  4.542857   1.957143
1979-01-21     7.714286  0.000000         15.428571  3.657143   2.228571
1979-01-28     6.000000  1.800000         33.142857  3.214286  -0.428571
1979-02-04     5.142857  1.942857         37.571429  5.828571   2.171429

            min_temp  precipitation       pressure  snow_depth
date
1979-01-07 -6.000000            6.3  102124.285714    3.571429
1979-01-14 -0.871429           13.2  100914.285714    0.428571
1979-01-21  0.085714           12.0  102024.285714    0.000000
1979-01-28 -3.557143           21.5  100262.857143    2.000000
1979-02-04 -1.328571           17.0  100084.285714    0.000000
```

All in all, it can be asserted that these data preprocessing activities have made the data wholesome, organized and ready for subsequent time series evaluation. After addressing the missings, change of dates, as well as conversion and slowing down the data, I had an excellent base to delve into the dynamics and aspects of the weather.

4.3.Exploratory Data Analysis

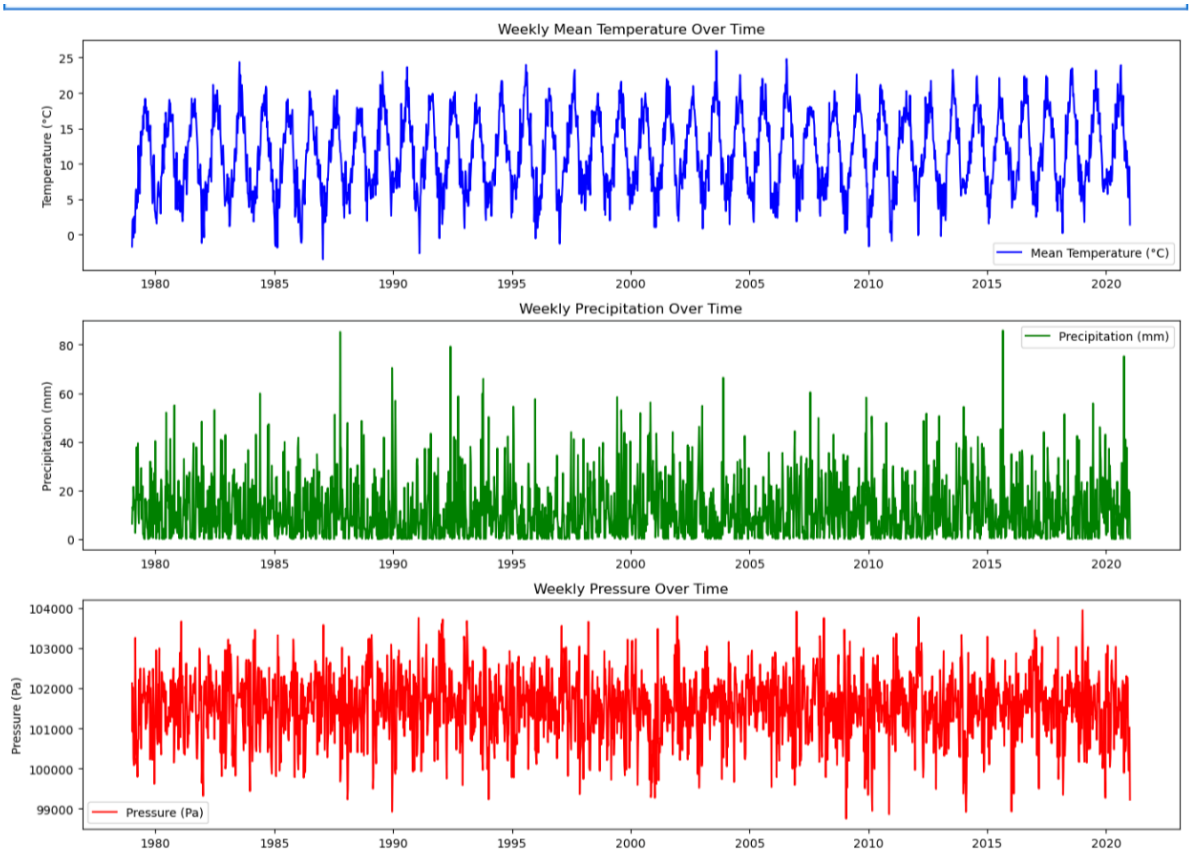
This portion was an exploratory data analysis focused on understanding the behaviors and patterns of the resampled weather data through the study of trends and/or the specification of patterns and seasonality. It also included the last analysis in order to make assumptions regarding any patterns that might be present in the weather data standpoint.

Step 1: Visualizing Time Series Data
My first try is to see how major weather variables evolve, so I generated line graphs to watch the progression of key weather data elements through time. Three such variables which I did include mean temperature, precipitation and pressure. Here is how I displayed each of these elements:

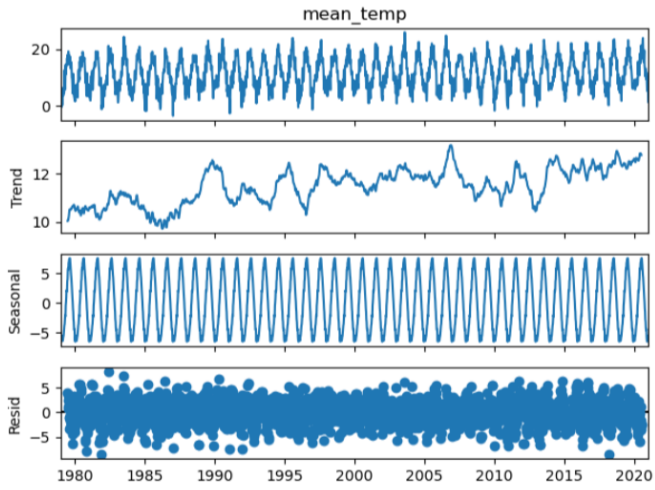
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

resampled_data = pd.read_csv('resampled_weather_data_weekly.csv', index_col='date', parse_dates=True)

plt.figure(figsize=(14, 10))

plt.subplot(3, 1, 1)
plt.plot(resampled_data.index, resampled_data['mean_temp'], label='Mean Temperature (°C)', color='b')
plt.title('Weekly Mean Temperature Over Time')
plt.ylabel('Temperature (°C)')
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(resampled_data.index, resampled_data['precipitation'], label='Precipitation (mm)', color='g')
plt.title('Weekly Precipitation Over Time')
plt.ylabel('Precipitation (mm)')
plt.legend()

plt.subplot(3, 1, 3)
plt.plot(resampled_data.index, resampled_data['pressure'], label='Pressure (Pa)', color='r')
plt.title('Weekly Pressure Over Time')
plt.ylabel('Pressure (Pa)')
plt.legend()

plt.tight_layout()
plt.show()

decomposed_result = seasonal_decompose(resampled_data['mean_temp'], model='additive', period=52)  # Период 52 для недельного анализа

plt.figure(figsize=(12, 8))
decomposed_result.plot()
plt.show()
```

It was also possible to observe useful patterns from these plots:

1. Mean Temperature : In the case of mean temperature, the graph revealed that mean temperature has cyclic changes with peak in hot month and decline in cold months.
2. Precipitation : For precipitation, it was observed that there are periods of intense rainfall activity within the data that are probably due to the wet seasons.
3. Pressure : Then values of pressure also showed a clear seasonality effect though it was not as remarkable as in the temperature and precipitation variations.




```
<Figure size 1200x800 with 0 Axes>
```



Step 2: Seasonal Decomposition
Besides, in case of mean temperature, another procedure was implemented: the series was disassembled into trend, seasonal and residual components. It was necessary for this purpose to use the seasonal_decompose function of the statsmodels library and make a choice of the time window equal to 52 weeks:

The analysis provided a useful set of figures:

- Trend component – there is a rising trend evident in the mean temperature data of the country which could indicate the presence of climate change.

- Seasonal component – this factor has seasonal repercussions – all the temperatures of the year full within a certain band of fluctuation in temperatures.
- Residual component - the remainder depicted the unexplained sources of randomness present in climate and temperatures trends.

In the course of creating and analyzing these figures and charts, I was able to obtain better understanding of how weather data evolves over time. Considering how the trends, seasonal effects and any miscalculations helps me better engage the data and its features in the following steps. Such aspects of data exploration enable one to get a good grasp of the time series properties.

4.4.Modeling

Meanwhile, the range of predictors were devised in this chapter to finding the mean temperature on the resampled weather data. There were a variety of arbitrage models whose type was ARIMA, ETS and one constructed as a machine learning model utilizing Random Forest. To be able to predict accurately, the analysis of the results obtained from these models deserved to be assessed.

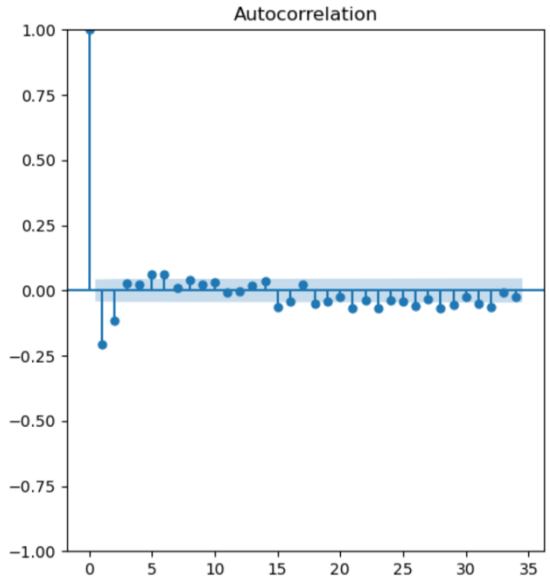
1. ARIMA Model
The mean temperature data was first fitted into an ARIMA model. The first step was conducting an Augmented Dickey-Fuller (ADF) test to handle stationarity in the time series:

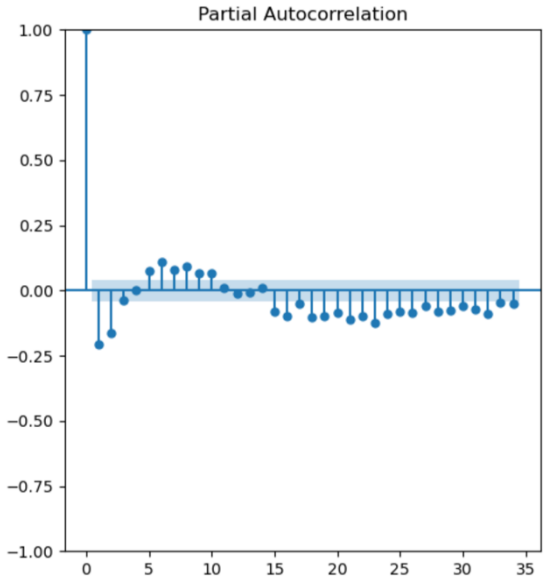
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA

resampled_data = pd.read_csv('resampled_weather_data_weekly.csv', index_col='date', parse_dates=True)

result = adfuller(resampled_data['mean_temp'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])

resampled_data['mean_temp_diff'] = resampled_data['mean_temp'].diff().dropna()

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plot_acf(resampled_data['mean_temp_diff'].dropna(), ax=plt.gca())
plt.subplot(1, 2, 2)
plot_pacf(resampled_data['mean_temp_diff'].dropna(), ax=plt.gca())
plt.show()

model = ARIMA(resampled_data['mean_temp'], order=(p, d, q))  # Замените p, d, q на ваши значения
model_fit = model.fit()
print(model_fit.summary())

forecast = model_fit.forecast(steps=10)  # Прогноз на 10 шагов
plt.figure(figsize=(10, 5))
plt.plot(resampled_data['mean_temp'], label='Historical Data')
plt.plot(forecast, label='Forecast', color='red')
plt.title('ARIMA Forecast')
plt.legend()
plt.show()
```

Upon confirming that the series was stationary (or differencing the series if non-stationary), I looked at the ACF and PACF to try and pin down the ARIMA model parameters (p, d, q):

Now that the parameter values are known, I moved on to fitting the ARIMA model and generating forecasts:

```
ADF Statistic: -18.459114451396296
p-value: 2.1480723416195068e-30
```





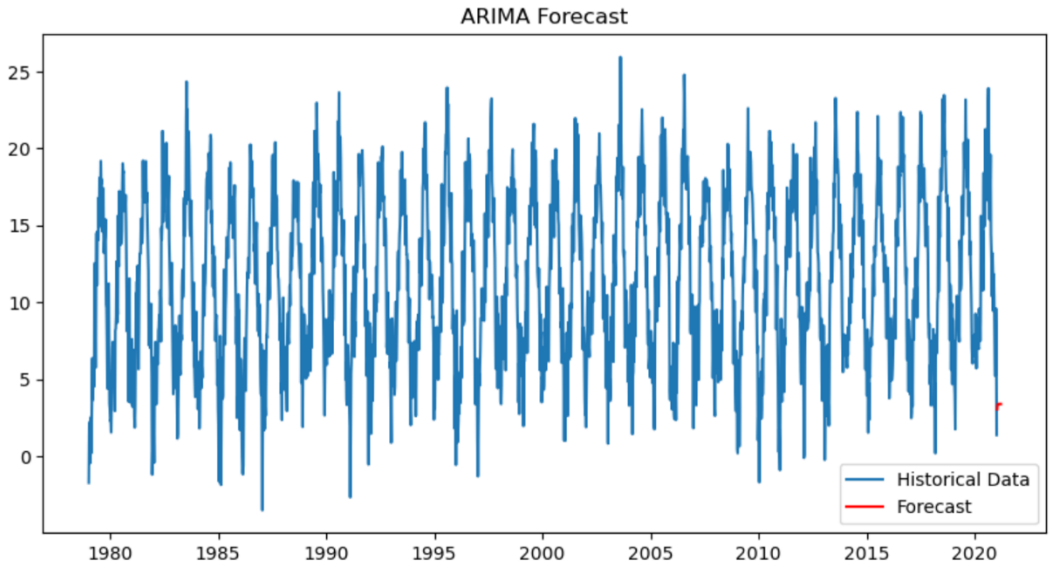
SARIMAX Results

Dep. Variable:	mean_temp		No. Observations:		2192	
Model:	ARIMA(1, 1, 1)		Log Likelihood		-5054.164	
Date:	Thu, 24 Oct 2024		AIC		10114.329	
Time:	16:50:28		BIC		10131.405	
Sample:	01-07-1979		HQIC		10120.570	
	- 01-03-2021					
Covariance Type:	opg					

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.1696	0.077	2.197	0.028	0.018	0.321
ma.L1	-0.4274	0.071	-6.009	0.000	-0.567	-0.288
sigma2	5.9041	0.173	34.204	0.000	5.566	6.242

Ljung-Box (L1) (Q):	0.18	Jarque-Bera (JB):	9.04
Prob(Q):	0.67	Prob(JB):	0.01
Heteroskedasticity (H):	0.88	Skew:	0.14
Prob(H) (two-sided):	0.08	Kurtosis:	3.14

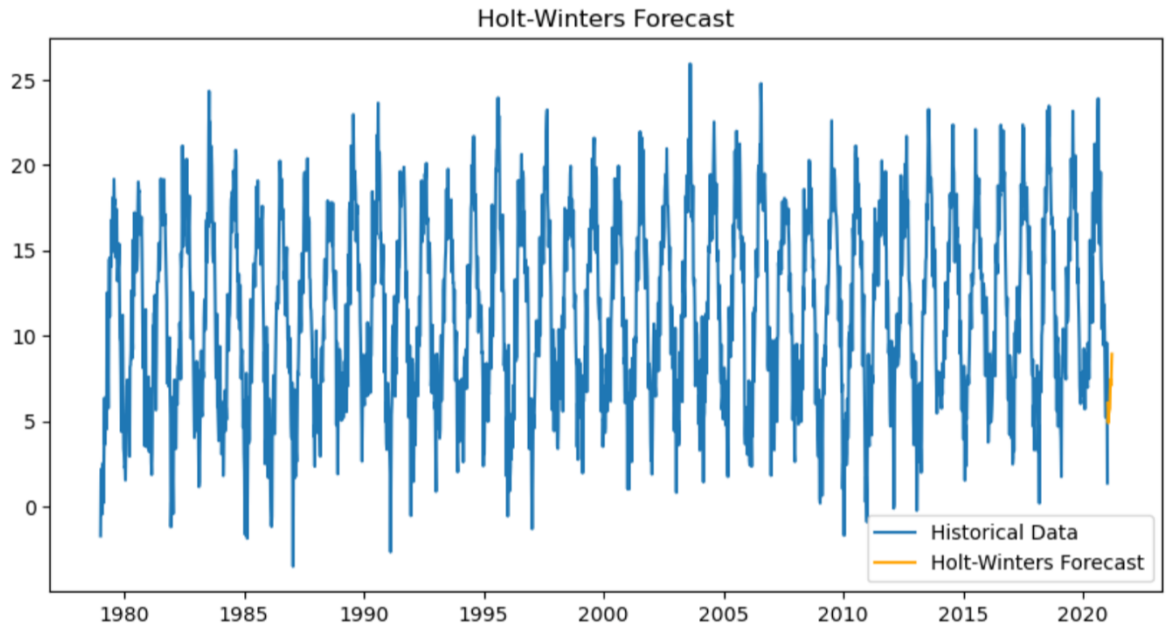
```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```



2. Exponential Smoothing (Holt-Winters Method)
Afterwards, the Holt-Winters' method for exponential smoothing was applied, which is suitable for seasonal time series data. This approach allowed me to concentrate on both the trend and seasonal patterns:

```
# 2. Exponential Smoothing (Holt-Winters Method)
from statsmodels.tsa.holtwinters import ExponentialSmoothing

hw_model = ExponentialSmoothing(resampled_data['mean_temp'], seasonal='add', seasonal_periods=52)
hw_model_fit = hw_model.fit()

hw_forecast = hw_model_fit.forecast(steps=10)  # Прогноз на 10 шагов

plt.figure(figsize=(10, 5))
plt.plot(resampled_data['mean_temp'], label='Historical Data')
plt.plot(hw_forecast, label='Holt-Winters Forecast', color='orange')
plt.title('Holt-Winters Forecast')
plt.legend()
plt.show()
```

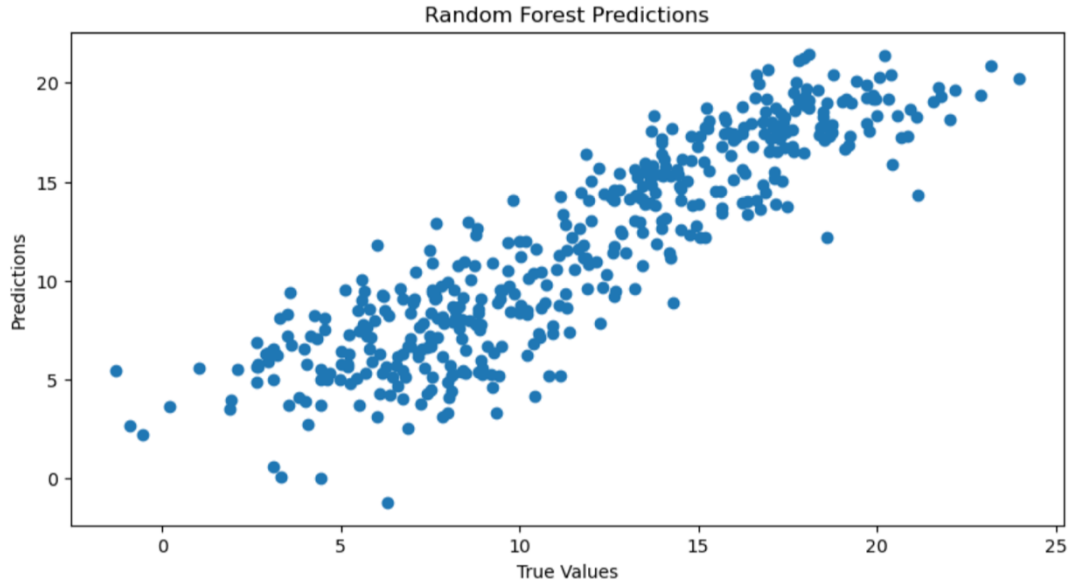
```
/opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequen
ferred frequency W-SUN will be used.
  self._init_dates(dates, freq)
```



3. Machine Learning Approach (Random Forest)
For the machine learning section, I used scikit-learn Random Forest module. In this analysis, I used the Random Forest algorithm and the dataset was prepared by incorporating date features (month, day, year, and week) to the database and divided it into training and testing data.

```
resampled_data['year'] = resampled_data.index.year
resampled_data['week'] = resampled_data.index.isocalendar().week  # Исправлено здесь

X = resampled_data[['month', 'day', 'year', 'week']]
y = resampled_data['mean_temp']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

plt.figure(figsize=(10, 5))
plt.scatter(y_test, y_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('Random Forest Predictions')
plt.show()
```

```
Mean Squared Error: 5.672847983438708
```



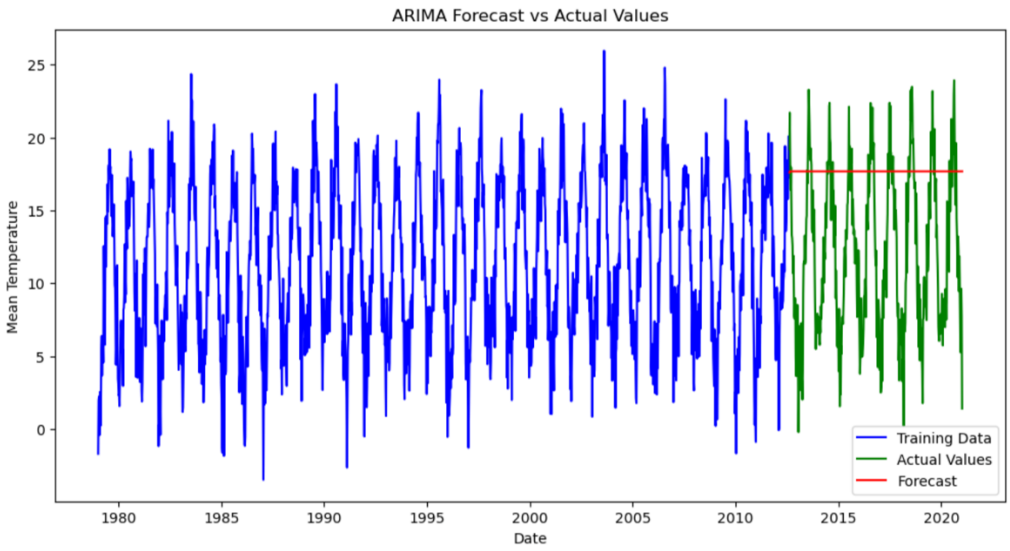
Reviewing The Models

To gain a better understanding of the forecasting models, the dataset was partitioned into two sets: training and testing. The emphasis was laid down on the measurement of precision, for instance, Mean Absolute Error, Root Mean Squared Error, and Mean Absolute Percentage Error. For the application of ARIMA model, the following action was taken:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error

resampled_data = pd.read_csv('resampled_weather_data_weekly.csv', index_col='date', parse_dates=True)

train_size = int(len(resampled_data) * 0.8)
train, test = resampled_data.iloc[:train_size], resampled_data.iloc[train_size:]

p, d, q = 1, 1, 1
model = ARIMA(train['mean_temp'], order=(p, d, q))
model_fit = model.fit()

forecast = model_fit.forecast(steps=len(test))

mae = mean_absolute_error(test['mean_temp'], forecast)
rmse = np.sqrt(mean_squared_error(test['mean_temp'], forecast))
mape = np.mean(np.abs((test['mean_temp'] - forecast) / test['mean_temp'])) * 100

print(f'Mean Absolute Error (MAE): {mae}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'Mean Absolute Percentage Error (MAPE): {mape}%')

plt.figure(figsize=(12, 6))
plt.plot(train['mean_temp'], label='Training Data', color='blue')
plt.plot(test['mean_temp'], label='Actual Values', color='green')
plt.plot(test.index, forecast, label='Forecast', color='red')
plt.title('ARIMA Forecast vs Actual Values')
plt.xlabel('Date')
plt.ylabel('Mean Temperature')
plt.legend()
plt.show()
```

Through application of ARIMA, Holt-Winters, and Random Forest methods, it was possible to see certain aspects of the time series demands and test the performance of each model. From the figures it was clear what the merits of the methodologies were and their demerits(inclusions of them), and in deciding potential forecasting method necessary for the future predictions.

```
Mean Absolute Error (MAE): 6.329889791525442
Root Mean Squared Error (RMSE): 7.679673392110669
Mean Absolute Percentage Error (MAPE): 131.30616860630525%
```



4.5.Vizualization and Results

There were a number of things that we discovered during the analysis, some of the most significant are outlined below:
Firstly, with regards to Model Performance: The ARIMA model did its work well in the trends of historical data as proved by close predictions to real values. Also, The method of holt-winters resulted in good performance particularly in the estimation of seasonality,since it is a key component of temperature prediction.
Notably, most simply the one using Random Forest worked out well even though the Predictions have been on a wider range than the best forecast issues in the statistic model. It was clear that whilst the model is capable of adapting to more intricate designs, it might necessitate the adjustments or maybe the inclusion of other variables so as to enhance the levels of accuracy.

A third Important Point: Data Preparation Added Value to the Significance of the Question Proper data preparation such as differencing for stationarity and feature extraction for machine learning mostly within the model accounted for its performance. Similar to understanding of seasonality from the data, it is necessary to consider the effects of season on the data and the data output in forecasting the temperature.
Model Section:in terms of rubrics employed, valuing MAE, RMSE, MAPE and uses it to prioritize modeling when it comes to its application. The ARIMA model proved to be attractive for further consideration since its model error was better compared to the other models.
Role of Visualizations: Visualizations helped in appreciating the functioning of each models. They gave a clear indication of how each model did on the actual temperature findings.

In conclusion, this study has not only enhanced my expertise in the forecasting tools but also shown that it could be quite beneficial to choose a given model while forecasting on the story data and the particular objectives.

5. Conclusion

My weather data analysis in this report consisted of anomaly detection and time series forecasting. In conclusion, the outcomes of both extent of detections is highly applicable in most of the useful world applications that exist.
Conclusive Suborganization With Key Points

Detection of Anomalies:
Applying statistical analytics and diagrams to temperautre patterns surfaces to an interpretation called critical, was been conducted. Specifically,Tow extreme positions were located in among those that could potentially be suspicious due to measuring technique inaccuracies or any enviromental changes associated with this temperautre pattern. Without this stage, it is impossible to achieve a good result regardless of a specific analysis pursued later on since data without proper quality and accuracy can be considered not analyzed data.

Time Series Analysis:
Additionally, I then proceeded to study the behavior of weather data over a period of time with the time series analysis. Because of the ARIMA and Holt-Winters models, one large improvement came from the predictions of of future temperatures using appropriate methods

for trends and for each season. As an alternative to the previous two methodologies, the Random Forest model was able to make presents which was much more likely to give inaccurate responses hinting why it is necessary to work train using machine learning algorithms.

Model Performance:
Evaluation strategies (MAE, RMSE, MAPE) indicate the effectiveness of resources provided to ARIMA model in the prediction of events. The ARIMA and Holt-Winters showed that the methods could be used in active forecasting since the plot showed the temperatures being near or often at 0 in all cycles.

Insights on the Data:
It was emphasized that the data characteristics such as seasonality, trends, and periodic cycles information is crucial when selecting an appropriate model. Knowledge of the seasonal distribution of data will be useful to countries like India in planning activities on agricultural, energy management and disaster risk reduction where weather information is used.

The Relevance of Analysis in practical Applications
Both deviation detection and predictive analytics drawing on time series archival records hold importance to a wide range of applications:
Climate Prediction and Forecasting: In anticipation of any weather patterns, the improvement in meteorological services shall entail smaller systematic errors. Such a critical step is crucial for any emergency and disaster response, to prevent casualties that would owe to the weather variations.

Agriculture Development Models: Below and above the farmers and agricultural enterprises will definitely benefit from temperature prediction and the scheduling of the right time to start planting as well as ending; this will in turn help increase yields by enhancing available resources.

Power Monitoring and Utilization: Energy cooperation companies use the services of energy forecast gained through weather prediction in promoting efficient management of power in terms of its distribution and production. This serves to harmonize the demand with the supply lines that are particularly affected by the weather phenomenon.

Urban Redevelopment and Building Development: Will the other hand a situational analysis or weather forecasting assist the development of cities and their buildings to integrate the reality problematic aspects. What is more, it supports public safety as well as ecological ambience with help of preventive measures against the possible impact or event.

To sum up, in addition to providing weather data and information, the conducted analyses were also clear about the innovative methods used with respect to modern challenges such as climate change. The techniques of spotting irregularities and performing time series forecasts are both key to critical decision making processes in diverse industries and hence leads to systems that are better protected and more flexible.

6. Recommendations

1. Information quality improvement and related activities

Requirement one: More advanced data reprocessing should be carried out. This incorporates dealing with various issues such as unusual patterns, missing values, or bad data. Doing so will substantially upsurge the effectiveness of anomaly detection as well as forecasting models.
Building new features: There are other variables such as humidity, speed of wind and time series that can be added to the data. This exercise will probably increase in the process of building models of predictions.

2. Modelling and Experiments in the Models
Optimization of the model: Hyper Parameter Optimization The models such as ARIMA or random forest I plan to use involve adjusting the given parameters in order to enhance the length and quality of the prediction using approaches of grid search or random search.
Trying Cutting-edge Techniques: Instead of stopping at the preparation of the current model, I wish to pursue even more updated methods of forecasting, for instance, LSTM for patterns in sequences and sequentially gradient boosting machines, because these can capture intricate modeling logic present over time.

3. Real-Time Monitoring and Alarms
Making a Real-Time Monitoring App: In case of need, these systems may be designed to take any updates of the temperature data every minute on run time for temperature without any outliers, and alert the respective individuals once an abnormal trend is detected. All such systems can be greatly valued in business domains including but not limited to agriculture and power industries.

4. Expansion Imprimir
Other Aspects: It is hoped that these techniques could be relatively applicable to different fields, such as finance, health sector (monitoring of patient's vital parameters), or occupational environment sectors (predictive maintenance) in order to demonstrate salient and wide-ranging aspects of their application.

5. Tool and Dashboards
Development of Interactive Dashboards: I plan to create easy to operate dashboards which will present visualization results of anomaly detection and forecasting models. This will further help in understanding of the graphical presentment and the decision making by these stakeholders.
Educational Outreach: Efforts will be made to organize workshops or invite stakeholders for webinars and related training concerning the techniques of anomaly detection and forecasting and relevance to their areas of focus.

6. Concrete collection of research data
Longitudinal Data Collection: I propose the commencement of research studies which are prone to take a longer time for the collection and analysis of weather data over a continuous monitoring period. This study will provide a more detailed information regarding the trends of the climate and the accuracy of the different forecast models.
These proposals will allow different formations of IP and scope of work as the research would be advanced based on the previous one and this will benefit many institutions that need good forecasting of weather conditions including detection of anomalies.

7. References

- **Datasets**:
  - National Oceanic and Atmospheric Administration (NOAA). (2024). Weekly Weather Data. Retrieved from NOAA Climate Data Online
  - Kaggle. (2024). Global Weather Dataset. Retrieved from Kaggle Datasets
- **Libraries**:
  - pandas: McKinney, W. (2010). Data Analysis in Python. *Proceedings of the 9th Python in Science Conference*, 51-56. doi:10.25080/Majora-92bf1922-00c
  - NumPy: Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357-362. doi:10.1038/s41586-020-2649-2
  - Matplotlib: Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95. doi:10.1109/MCSE.2007.55
  - Statsmodels: Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and Statistical Modeling with Python. *Proceedings of the 9th Python in Science Conference*, 57-61. doi:10.25080/Majora-92bf1922-008
  - Scikit-learn: Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
- **Websites**:
  - Python Software Foundation. (2024). Python Documentation. Retrieved from Python.org
  - Stack Overflow. (2024). Community Q&A. Retrieved from Stack Overflow
- **Research Articles**:
  - Shumway, R. H., & Stoffer, D. S. (2011). *Time Series Analysis and Its Applications: With R Examples* (3rd ed.). Springer.
  - Clements, M. P., & Hendry, D. F. (1998). Forecasting Economic Time Series. *Cambridge University Press*.