1/25/2024

Projekt Hulumtues

Analiza e të dhënave Meteorologjike në Kosovë

Bledi Buduri
PROF. ASS. DR.ZIRIJE HASANI

Bledi Buduri

Fakulteti i Shkencave Kompjuterike - Drejtimi TIT

Universiteti i Prizrenit "Ukshin Hoti"- Prizren,Kosovë

25/01/2024

Përmbajtja

1.Hyrje rreth Projektit 3

2.Krijimi i Datasetit të kompletuar 4

3.Paraqitja e të dhënave nga Dataseti dhe llogaritja e vlerave të rëndësishme 8

3.1 Resampling sipas Muajit dhe Vitit 12

4.Vizualizimi dhe Analiza e Korrelacionit të Shumëfishtë 15

4.1. Vizualizimi i korrelacionit midis të dhënave: 16

4.2 Paraqitja e Regresionit ndërmjet Lagështisë dhe Apperent Temperature 18

4.3 Vizualizime përmbyllëse të pjesës së pare 19

5. Vizualizimi i rezultateve të parashikimeve SARIMA 23

5.1. Vizualizimi i Rezultateve për Datasetin Temperatura 23

5.2. Vizualizimi i Rezultateve për Datasetin Humidity(Lageshtia) 26

5.3. Vizualizimi i Rezultateve për Datasetin WindSpeed(Shpejtesia e erës) 29

5.4. Vizualizimi i rezultateve për Datasetin AirPreasure(Presioni i Ajrit) 31

6.Konkluzioni 34

Referencat 35

1.Hyrje rreth Projektit

Projekti ynë hulumtues ka për qëllim të kryejë analizën e të dhënave meteorologjike në Kosovë për periudhën kohore nga viti 2017 deri në vitin 2022. Për këtë qëllim, do të përdorim të dhënat e disponueshme të cilat i kemi pranuar. Analizat tona do të përqëndrohen në lidhjen dhe ndikimin reciprok të temperaturës, lagështisë , presionit të ajrit dhe shpejtësisë së erës.

Detyra e parë e këtij projekti do të përfshijë analizën e korrelacionit të shumëfishtë midis temperaturës, lagështisë, shpejtësisë së erës dhe presionit të ajrit duke përdorur gjuhën programuese Python. Kjo analizë do të na ndihmojë të identifikojmë marrëdhëniet komplekse dhe ndërveprime midis këtyre parametrave meteorologjikë, duke shtuar në këtë mënyrë një kuptim në kuptimin e dinamikave të kushteve atmosferike në Kosovë gjatë periudhës kohore të studiuar.

Detyra e dytë e projektit përfshin vizualizimin e rezultateve të parashikimeve duke përdorur algoritmin SARIMA për temperaturë, lagështi, shpejtësi të erës dhe presionit të ajrit. Përdorimi i këtij algoritmi do të na mundësojë të parashikojmë zhvillimet e mundshme të këtyre parametrave në të ardhmen duke analizuar trendet dhe variacionet e tyre. Nëpërmjet vizualizimeve të përpunuara grafikisht, do të paraqesim në një mënyrë të qartë dhe të kuptueshme rezultatet e parashikimeve të këtyre parametrave.

Ky projekt synon të sjellë një kontribut në kuptimin e kushteve meteorologjike në Kosovë në një periudhë kohore të caktuar, duke përdorur një kombinim të analizave statistikore dhe algoritmave të avancuara për parashikim. Me shpresën që rezultatet e këtij projekti do të ofrojnë një bazë të mirë për kuptimin e dinamikave të ndryshme atmosferike dhe ndihmojnë në hartimin e politikave dhe vendimeve të bazuara në dije për menaxhimin e kushteve meteorologjike në Kosovë.

2.Krijimi i Datasetit të kompletuar

Projekti ynë i hulumtimit ka për qëllim analizën e të dhënave meteorologjike në Kosovë nga viti 2017 deri në vitin 2022, duke përdorur një gamë të gjerë të teknikave analitike dhe algoritmave për parashikim. Për të arritur në rezultatet e dëshiruara, është e rëndësishme importimi i disa librave kyçe në fillim të projektit.

Për manipulimin dhe analizën e të dhënave, kemi zgjedhur të përdorim *pandas*, një librari e specializuar për të punuar me të dhënat tabelore. *Numpy* është importuar për të mundësuar operacione numerike efikase. Për krijimin e vizualizimeve tërheqëse dhe informativ, kemi përdorur *Matplotlib* në kombinim me *Seaborn*, e cila sjell stilizime të përmirësuara për grafikat.

Për të siguruar një prezantim më të pastër të vizualizimeve, kemi ndaluar shfaqjen e mesazheve të paralajmërimit duke përdorur warnings. Një tjetër element kyç i prezantimit është përdorimi i stilizimit të Seaborn me `*sns.set(style="darkgrid")*`, duke ofruar një pamje më tërheqëse dhe të kuptueshme për lexuesin.

```
#Import neccessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
sns.set(style="darkgrid")
```

Të dhënat që do të marrim për temperaturën, lagështin, presioni ajrit, dhe shpejtësia e erës do të jenë nga katër fajlle të ndryshme dhe secili prej tyre kishte distanca kohore të ndryshme. Kështu që në vazhdim të kodit do të shohim fillimisht se si do të bëjmë leximin e këtyre fajlleve që janë të formës CSV dhe ruajtjen e tyre në DataFrame.

```
#Bejme leximin e Dataseteve
#Temperature
df1 = pd.read_csv(r'C:\Users\Admin\Desktop\Projekti
Hulumtues\temperature.csv')
#Humidity
df2 = pd.read_csv(r'C:\Users\Admin\Desktop\Projekti Hulumtues\humidity.csv')
#AirPreasure
df3 = pd.read_csv(r'C:\Users\Admin\Desktop\Projekti
Hulumtues\airpreasure.csv')
#WindSpeed
df4 = pd.read_csv(r'C:\Users\Admin\Desktop\Projekti Hulumtues\windspeed.csv')
```

Siç edhe e cekëm mësipër pas këtij veprimi do të bëjmë filtrimin e të dhënave nga secili prej dataseteve duke përcaktuar kështu periudhën kohore nga viti 2017 deri në vitin 2022. (Në vazhdim do të shohim vetëm filtrimin e të dhënave vetëm nga dataseti ‘AirPressure’ – Presioni i ajrit.

Dhe ngjashëm veprohet me secilin nga datasetet). Ajo që është e rëndësishme të dihet para bërjes së këtij veprimi është të bëhet kontrollimi se prej kur fillon dhe deri kur mbaron periudha kohore në atë fajll.

```
#AirPreasure
df3_2017_2022 = df3[(df3['datetime'] >= '01.01.2017 00:00:00') &
(df3['datetime'] <= '31.12.2022 23:00:00')]
```

Kodi në vazhdim është përdorur për të konvertuar kolonën e datës (datetime) të katër DataFrame-ve (`df1_2017_2022`, `df2_2017_2022`, `df3_2017_2022`, `df4_2017_2022`) në formatin e datetime duke përdorur bibliotekën Pandas në Python.

Le të shpjegojmë secilin rresht të kodit:

```
#Convert date column to datetime
df1_2017_2022['datetime'] = pd.to_datetime(df1_2017_2022['datetime'],
utc=True, errors='coerce')
df2_2017_2022['datetime'] = pd.to_datetime(df2_2017_2022['datetime'],
utc=True, errors='coerce')
df3_2017_2022['datetime'] = pd.to_datetime(df3_2017_2022['datetime'],
utc=True, errors='coerce')
df4_2017_2022['datetime'] = pd.to_datetime(df4_2017_2022['datetime'],
utc=True, errors='coerce')
```

- pd.to_datetime - Kjo metodë e Pandas është përdorur për të konvertuar kolonën e datës në formatin e datetime.
- Parametri i parë është kolona e datës që po konvertohet (`dfX_2017_2022['datetime']`), ku `X` është numri i DataFrame (1, 2, 3, 4).
- Parametri i dytë (`utc=True`) tregon që të dhënat e datetime do të trajtohen si të kohës universal të kohezonit (UTC).
- Parametri i tretë (`errors='coerce'`) tregon që në rast të ndonjë gabimi gjatë konvertimit, vlerat të cilat nuk janë të konvertueshme do të zëvendesohen me `NaT` (Not a Time).

Kjo procedurë është e rëndësishme kur do të përdorim kolonën e datës për operacione kohore, analiza të ndryshme dhe vizualizime. Konvertimi në formatin e datetime mund të lehtësojë shumë manipulimet dhe analizat e mëtejshme të të dhënave.

Kjo pjesë e kodit është përdorur për të krijuar një dataset të përbashkët apo të themi më mire të plotësuar duke përfshirë një gamë të plotë të datave për periudhën kohore nga viti 2017 deri në vitin 2022.

Shpjegojmë secilin rresht të kodit:

```
# Generate complete date range
start_date = '2017-01-01 00:00:00'
end_date = '2022-12-31 23:00:00'
date_range = pd.date_range(start=start_date, end=end_date, freq='H')
```

1. ***Krijimi i gamës së plotë të datave:*** Në këtë rresht, një gamë e plotë e datave është krijuar nga data fillestare ('2017-01-01 00:00:00') deri në datën përfundimtare ('2022-12-31 23:00:00') me një frekuencë orare ('H') . Rezultati është ruajtur në variablën `date_range`.

```
# Create DataFrames with the complete date range
complete_df1 = pd.DataFrame({'datetime': date_range})
```

2. ***Krijimi i DataFrame me gamën e plotë të datave****:* Në këtë rresht, një DataFrame i ri është krijuar duke përfshirë kolonën 'datetime' dhe vlerat nga një gamë e plotë e datave (`date_range`). Rezultati është ruajtur në variablën `complete_df1`.

```
# Convert datetime to UTC
complete_df1['datetime'] = complete_df1['datetime'].dt.tz_localize('UTC')
```

3. ***Konvertimi i datetime në kohëzonin UTC****:* Në këtë rresht, kolona e datetime në DataFrame është konvertuar në kohëzonin UTC duke përdorur metoden `tz_localize('UTC')`.

```
# Perform left join to include all dates
merged_dataset = pd.merge(complete_df1, df1_2017_2022, on='datetime',
how='left')
merged_dataset = pd.merge(merged_dataset, df2_2017_2022, on='datetime',
how='left', suffixes=('_Temperature', '_Humidity'))
```

4. ***Bashkimi i dataseteve me gamën e plotë të datave****:* Në këto rreshta, është kryer një bashkim i të gjitha dataseteve (`df1_2017_2022`, `df2_2017_2022`, `df3_2017_2022`, `df4_2017_2022`) me `complete_df1` duke përdorur një left join bazuar në kolonën 'datetime'. Ky bashkim siguron që të gjitha datat nga gama e plotë janë përfshirë në datasetin përfundimtar `merged_dataset`. Suffixes janë shtuar në kolonat përkatëse për të dalluar vlerat e kolonave të njëjta nga datasetet e ndryshme.

Kështu, kodi përfundimtar ka si rezultat një dataset të përbashkët që përmban të dhënat të të gjitha kategorive (temperaturë, lagështi, presion i ajrit, shpejtësi e erës) për një periudhë kohore të plotë nga '2017-01-01 00:00:00' deri në '2022-12-31 23:00:00'.

```
#To save the merged DateFrame to a new CSV file
merged_dataset.to_csv('datasetprova1.csv', index=False)

# Rename columns
df.rename(columns={'T_mu': 'Temperature', 'H_mu_Humidity': 'Humidity',
'H_mu_AirPressure': 'AirPressure', 'H_mu': 'WindSpeed'}, inplace=True)
```

Dhe meqë emërtimet nuk janë në formën e duhur të tyre ato do t'i riemërojmë dhe do t'i ruajmë në datasetin e fundit të kompletuar.

Kurse ajo që na nevojitet për të kompletuar datasetin përfunimtarë është edhe llogaritja e Apparent Temperature e cila është e bazuar në temperatur dhe lagështi. Këtë llogaritje do të bëjmë përmes një funksioni të cilin pastaj do të aplikojmë atë në një DataFrame në mënyrë që këto rezultate t'i paraqesim në një kolonë si pjesë e datasetit të kompletuar.

```
def calculate_apparent_temperature( temperature, humidity):
    # Constants for the heat index formula
    c1 = -42.379
    c2 = 2.04901523
    c3 = 10.14333127
    c4 = -0.22475541
    c5 = -6.83783e-03
    c6 = -5.481717e-02
    c7 = 1.22874e-03
    c8 = 8.5282e-04
    c9 = -1.99e-06

    # Calculate the heat index
    heat_index = c1 + (c2 * temperature) + (c3 * humidity) + (c4 *
temperature * humidity) + (c5 * temperature ** 2) + (c6 * humidity ** 2) +
(c7 * temperature ** 2 * humidity) + (c8 * temperature * humidity ** 2) + (c9
* temperature ** 2 * humidity ** 2)

    return heat_index

# Apply the function to create a new column 'ApparentTemperature'
df['ApparentTemperature'] = df.apply(lambda row:
calculate_apparent_temperature(row['Temperature'], row['Humidity']), axis=1)
```

Siç edhe shihet nga kodi sipër pjesa parë e saj paraqet llogaritjen e Apparent Temperature kurse në pjesën e dytë të saj është paraqitur shfrytëzimi i apply dhe lambda që shërben për të kërkuar të dhënat për Temperature dhe Humidity.

```
# Save the updated DataFrame to a new CSV file
df.to_csv('updated.csv', index=False)
```

Dhe pjesa e fundit e këtij kodi paraqet ruajtjen e të dhënave me një dataset të kompletuar dhe ruajtjen e tij si nje fajll CSV.

3.Paraqitja e të dhënave nga Dataseti dhe llogaritja e vlerave të rëndësishme

Ajo që është me rëndësi brenda kornizave të paraqitjes së datasetit punues është edhe paraqitja e të dhënave duke përdorur disa funksione bazike të bashkëngjitura me rezultatet e tyre:

```
#Shfaq vetem rreshtat 5 rreshtat e pare
print(df.head())
```

OUTPUT:

```
                   datetime  Temperature  Humidity  AirPressure  WindSpeed  \
0  2017-01-01 00:00:00+00:00       -7.674    92.576      984.771        NaN
1  2017-01-01 01:00:00+00:00       -7.914    94.203      984.308        NaN
2  2017-01-01 02:00:00+00:00       -8.612    95.751      983.977        NaN
3  2017-01-01 03:00:00+00:00       -8.928    95.533      983.845        NaN
4  2017-01-01 04:00:00+00:00       -8.299    92.854      983.249        NaN

   ApparentTemperature
0           520.001068
1           523.860131
2           533.495794
3           537.631070
4           528.217972
```

```
#Shfaq 5 rreshtat e fundit
print(df.tail())
```

OUTPUT:

```
                       datetime  Temperature  Humidity  AirPressure  \
52579  2022-12-31 19:00:00+00:00          NaN       NaN      982.388
52580  2022-12-31 20:00:00+00:00          NaN       NaN      983.050
52581  2022-12-31 21:00:00+00:00          NaN       NaN      983.183
52582  2022-12-31 22:00:00+00:00          NaN       NaN      983.183
52583  2022-12-31 23:00:00+00:00          NaN       NaN      983.778

       WindSpeed  ApparentTemperature
52579        NaN                  NaN
52580        NaN                  NaN
52581        NaN                  NaN
52582        NaN                  NaN
52583        NaN                  NaN
```

Arsyeja pse në kolona ka përmbajtje të vlerës NaN është për shkakë se ka vlera jo të plotësuara në dataset që në të njëjtën kohë vështirson punën për rezultate më të mira.

Në kodin në vazhdim kemi përdorur komandën shape e cila si rezultat paraqet numrin e rreshtave dhe të kolonave në DataFrame kurse komanda describe shfaq një analizë statistikore të kolonave numerike të DataFrame. Për secilën kolonë numerike, do të shfaqen vlera statistikore si: numri i vlerave, vlera mesatare (mean), devijimi standard (standard deviation), vlera minimale (minimum), përqindja e 25-të, mediana (përqindja e 50-të), përqindja e 75-të, dhe vlera maksimale (maximum). Ky rezultat ofron një pasqyrë të shpejtë të tendencave dhe variablave të datasetit.

```
print(df.shape)
print(df.describe())
```

OUTPUT:

```
(52584, 6)
        Temperature      Humidity   AirPressure     WindSpeed  \
count  16816.000000  14032.000000  47331.000000  15187.000000
mean       9.374791     76.757756    968.696981     13.957145
std        9.753884     19.405988      6.782307    111.586410
min      -22.385000     19.000000    940.626000      0.000000
25%        2.500000     58.790000    964.386000      0.520000
50%        7.800000     80.000000    968.291000      0.950000
75%       13.700000     96.000000    972.792000      1.700000
max       30.000000    100.240000    997.744000    999.990000

       ApparentTemperature
count         14032.000000
mean            320.491490
std              93.804073
min             107.008824
25%             264.978550
50%             331.402595
75%             385.545805
max             704.871347
```

3.1. Llogaritja e vlerave të korrelacionit

Pjesa në vazhdim e kodit është përdorur për të llogaritur dhe shfaqur matricën e korrelacionit midis kolonave numerike të DataFrame (**df**).

```
# Exclude 'datetime' column from correlation calculation
columns_for_correlation = df.columns.difference(['datetime'])

# Calculate correlation
correlation_matrix = df[columns_for_correlation].corr(method='pearson')
```

```
# Print correlation matrix
print(correlation_matrix)
```

```
                     AirPressure  ApparentTemperature  Humidity  Temperature  \
AirPressure             1.000000             0.296189  0.097888    -0.262360
ApparentTemperature     0.296189             1.000000  0.767427    -0.968386
Humidity                0.097888             0.767427  1.000000    -0.678790
Temperature            -0.262360            -0.968386 -0.678790     1.000000
WindSpeed              -0.001076             0.012495  0.004940    -0.012122

                     WindSpeed
AirPressure          -0.001076
ApparentTemperature   0.012495
Humidity              0.004940
Temperature          -0.012122
WindSpeed             1.000000
```

Kodi në vazhdim bën mbledhjen e vlerave null në secilën nga kolonat e atij dataseti kurse pjesa e dytë e kodit tregon në qoftë se kolonat kanë vlera null të na jap rezultat True.

```
# Print the count of null values for each column
print(df.isnull().sum())
# Check if any column has null values
print(df.isna().any())
```

OUTPUT:

```
datetime                0
Temperature             35768
Humidity                38552
AirPressure             5253
WindSpeed               37397
ApparentTemperature     38552
dtype: int64
datetime                False
Temperature              True
Humidity                 True
AirPressure              True
WindSpeed                True
ApparentTemperature      True
```

Pjesa e kodit në vazhdim kryen disa operacione mbi kolonën e datës, shfaq informacione rreth DataFrame duke përdorur `info()`, dhe krijon një DataFrame të ri duke normalizuar një kolonë të dhënash.

1. ***Konvertimi i kolonës së datës në formatin datetime:*** Në këtë rresht, kolona e datës (`'datetime'`) është konvertuar në formatin datetime duke përdorur metoden `to_datetime` të bibliotekës Pandas. Parametri `utc=True` tregon se koha do të trajtohet si në kohëzonin UTC, dhe `errors='coerce'` do të zëvendësojë vlerat jovalide me *`NaT` (Not a Time).*
2. ***Shfaqja e informacioneve për DataFrame:*** Në këtë rresht, janë shfaqur informacione të përgjithshme rreth DataFrame duke përdorur metoden `info()`. Kjo përfshin numrin e rreshtave, numrin e kolonave, tipin e të dhënave për secilën kolonë, dhe numrin e vlerave të munguara.
3. ***Krijimi i një DataFrame të ri me normalizimin e Apparent Temperature:*** Në këtë rresht, një DataFrame i ri është krijuar duke kopjuar të dhënat nga DataFrame origjinale (`df`) nëpërmjet `copy()`. Kolona 'ApparentTemperature' është normalizuar duke përdorur formulën `(vlera - vlera_min) / (vlera_max - vlera_min)`, ku `vlera_min` është vlera minimale dhe `vlera_max` është vlera maximale të kolonës 'ApparentTemperature'. Ky normalizim përbën një shkallë nga 0 deri në 1, ku vlera 0 përfaqëson minimumin dhe vlera 1 përfaqëson maksimumin të kolonës 'ApparentTemperature'.
   Rezultati është shfaqur duke përdorur `print(df_scaled.head())` për të verifikuar ndryshimet në DataFrame.

```
#Convert date column to datetime
df['datetime'] = pd.to_datetime(df['datetime'], utc=True, errors='coerce')

df.info()

#Creating a new dataframe and normalising Apparent Temperature
df_scaled = df.copy()
df_scaled['ApparentTemperature'] = (df_scaled['ApparentTemperature'] -
df_scaled['ApparentTemperature'].min()) /
(df_scaled['ApparentTemperature'].max() -
df_scaled['ApparentTemperature'].min())
print(df_scaled.head())
```

OUTPUT:

```
dtype: bool
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52584 entries, 0 to 52583
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   datetime            52584 non-null  datetime64[ns, UTC]
 1   Temperature         16816 non-null  float64
 2   Humidity            14032 non-null  float64
 3   AirPressure         47331 non-null  float64
 4   WindSpeed           15187 non-null  float64
 5   ApparentTemperature  14032 non-null  float64
dtypes: datetime64[ns, UTC](1), float64(5)
memory usage: 2.4 MB
```

```
                   datetime  Temperature  Humidity  AirPressure  WindSpeed  \
0 2017-01-01 00:00:00+00:00       -7.674    92.576      984.771        NaN
1 2017-01-01 01:00:00+00:00       -7.914    94.203      984.308        NaN
2 2017-01-01 02:00:00+00:00       -8.612    95.751      983.977        NaN
3 2017-01-01 03:00:00+00:00       -8.928    95.533      983.845        NaN
4 2017-01-01 04:00:00+00:00       -8.299    92.854      983.249        NaN

   ApparentTemperature
0             0.690781
1             0.697236
2             0.713353
3             0.720270
4             0.704525
```

3.1 Resampling sipas Muajit dhe Vitit

Kjo pjesë e kodit që do të shohim në vazhdim kryen disa operacione të rëndësishme për analizën kohore të të dhënave.

- ***Resampling sipas muajit dhe vitit:*** Në këtë segment, të dhënat janë "resampled" sipas muajit dhe vitit duke përdorur metoden `resample` nga Pandas, ku pjesa `set_index('datetime')` vendos kolonën 'datetime' si indeks të DataFrame, pastaj `resample('M')` përdoret për të ndryshuar frekuencën e të dhënave në një frekuencë mujore (M) duke marrë vlerat mesatare për çdo muaj dhe `mean()` është përdorur për të llogaritur mesataren e vlerave për secilën kolonë për secilin muaj ose vit.
- ***Rivendosja e indeksit të DataFrame:*** Pas resampling, indeksat janë ndryshuar në një kolonë të zakonshme në DataFrame. Për të përshtatur këtë ndryshim, `reset_index(inplace=True)` është përdorur për të rivendosur indeksin, duke bërë indeksin një kolonë të thjeshtë numerike.

```
# Resample according to month and year
monthly_data = df.set_index('datetime').resample('M').mean()
yearly_data = df.set_index('datetime').resample('Y').mean()
df_monthly_scaled = df_scaled.set_index('datetime').resample('M').mean()


# Reset the index
monthly_data.reset_index(inplace=True)
yearly_data.reset_index(inplace=True)
df_monthly_scaled.reset_index(inplace=True)

# Print the first few rows of the resulting DataFrame
print(monthly_data.head())

#e morrem pjesen e tail per me shiku nese ka ndryshime ne resultate
print(monthly_data.tail())

# Print the first few rows of the resulting DataFrame
print(yearly_data.head())
```

OUTPUT:

```
                  datetime  Temperature   Humidity  AirPressure  WindSpeed  \
0 2017-01-31 00:00:00+00:00    -3.887036  81.650934   974.743978  13.957145
1 2017-02-28 00:00:00+00:00     9.374791  76.757756   973.801123  13.957145
2 2017-03-31 00:00:00+00:00     9.374791  76.757756   971.091014  13.957145
3 2017-04-30 00:00:00+00:00     9.374791  76.757756   969.639894  13.957145
4 2017-05-31 00:00:00+00:00     9.374791  76.757756   969.869053  13.957145

   ApparentTemperature
0           451.639255
1           320.491490
2           320.491490
3           320.491490
4           320.491490
                   datetime  Temperature   Humidity  AirPressure  WindSpeed  \
67 2022-08-31 00:00:00+00:00     9.374791  76.757756   965.209757  13.638232
68 2022-09-30 00:00:00+00:00     9.374791  76.757756   965.995878  13.957145
69 2022-10-31 00:00:00+00:00    12.575378  76.757756   971.651664  13.882107
70 2022-11-30 00:00:00+00:00     7.736121  76.757756   967.400930  13.937760
71 2022-12-31 00:00:00+00:00     6.543769  76.757756   973.287805  13.863347

    ApparentTemperature
67            320.49149
68            320.49149
69            320.49149
70            320.49149
71            320.49149
                  datetime  Temperature   Humidity  AirPressure  WindSpeed  \
0 2017-12-31 00:00:00+00:00     7.905485  77.629710   970.590491  13.957145
1 2018-12-31 00:00:00+00:00    11.509082  74.015898   967.932359  10.151682
2 2019-12-31 00:00:00+00:00    11.304162  75.185179   968.221476   9.406652
3 2020-12-31 00:00:00+00:00     8.408032  79.019507   968.652196   7.983543
4 2021-12-31 00:00:00+00:00     8.537250  77.932291   967.564699   8.777474

   ApparentTemperature
0           333.730708
1           303.919354
2           305.563774
3           331.083697
4           328.130898
```

Kjo pjesë e kodit është përdorur për të krijuar një DataFrame të ri që përmban temperaturën maksimale dhe minimale të ndjesuar për secilin vit. Le të shpjegojmë çdo rresht të kodit:

```
df1 = df[['datetime','ApparentTemperature']]
```

1. ***Krijimi i një DataFrame të ri me kolonat e përzgjedhura:***
   Në këtë rresht, një DataFrame i ri është krijuar duke përzgjedhur vetëm kolonat 'datetime' dhe 'ApparentTemperature' nga DataFrame origjinale (`df`).

```
#Maximum and minimum temperature per year
max_temp = df1.set_index('datetime').resample('Y').max()[1:]
min_temp = df1.set_index('datetime').resample('Y').min()[1:]
```

2. ***Llogaritja e temperaturës maksimale dhe minimale sipas vitit:*** Në këto dy rreshta, temperaturat maksimale dhe minimale janë llogaritur duke përdorur metoden `resample` për të grupuar të dhënat sipas vitit (`'Y'`) dhe duke përdorur funksionet `max` dhe `min` për të gjetur vlerat maksimale dhe minimale për secilin vit.

```
max_temp.reset_index(inplace=True)
min_temp.reset_index(inplace=True)
```

3. ***Rivendosja e indeksit të DataFrame:*** Pas llogaritjes së temperaturave maksimale dhe minimale, indeksat janë ndryshuar. Për të rivendosur indeksin në një kolonë numerike, `reset_index(inplace=True)` është përdorur.

```
max_temp.rename(columns={'ApparentTemperature': 'Max Apparent Temperature
(C)'}, inplace=True)
min_temp.rename(columns={'ApparentTemperature': 'Min Apparent Temperature
(C)'}, inplace=True)
```

4. ***Rivendosja e emrave të kolonave:*** Pas rivendosjes së indeksit, emrat e kolonave janë ndryshuar për të përshtatur përmbajtjen e tyre. Përmes `rename`, emrat e kolonave janë ndryshuar në "Max Apparent Temperature (C)" dhe "Min Apparent Temperature (C)".

```
min_max_temp = pd.merge(max_temp, min_temp, how='inner',  on='datetime')
min_max_temp['datetime'] = min_max_temp['datetime'].dt.year
```

5. ***Bashkimi i DataFrame-ve dhe ndryshimi i kolonës 'datetime':*** Në këtë rresht, DataFrame-të `max_temp` dhe `min_temp` janë bashkuar përmes një join të brendshëm (`inner join`) bazuar në kolonën 'datetime'. Pastaj, kolona 'datetime' është konvertuar në vit duke përdorur `dt.year` për të hequr detajet e kohës dhe për të mbetur vetëm me vitin.

Rezultati përfundimtar është shfaqur duke përdorur `print(min_max_temp)`. Ky DataFrame përmban temperaturën maksimale dhe minimale të ndjesuar për secilin vit, ku emrat e kolonave janë "Max Apparent Temperature (C)" dhe "Min Apparent Temperature (C)".

	datetime	Max Apparent Temperature (C)	Min Apparent Temperature (C)
0	2018	534.514761	107.008824
1	2019	515.045103	136.292835
2	2020	533.448576	131.360964
3	2021	547.677975	146.151370
4	2022	320.491490	320.491490

4. Vizualizimi dhe Analiza e Korrelacionit të Shumëfishtë

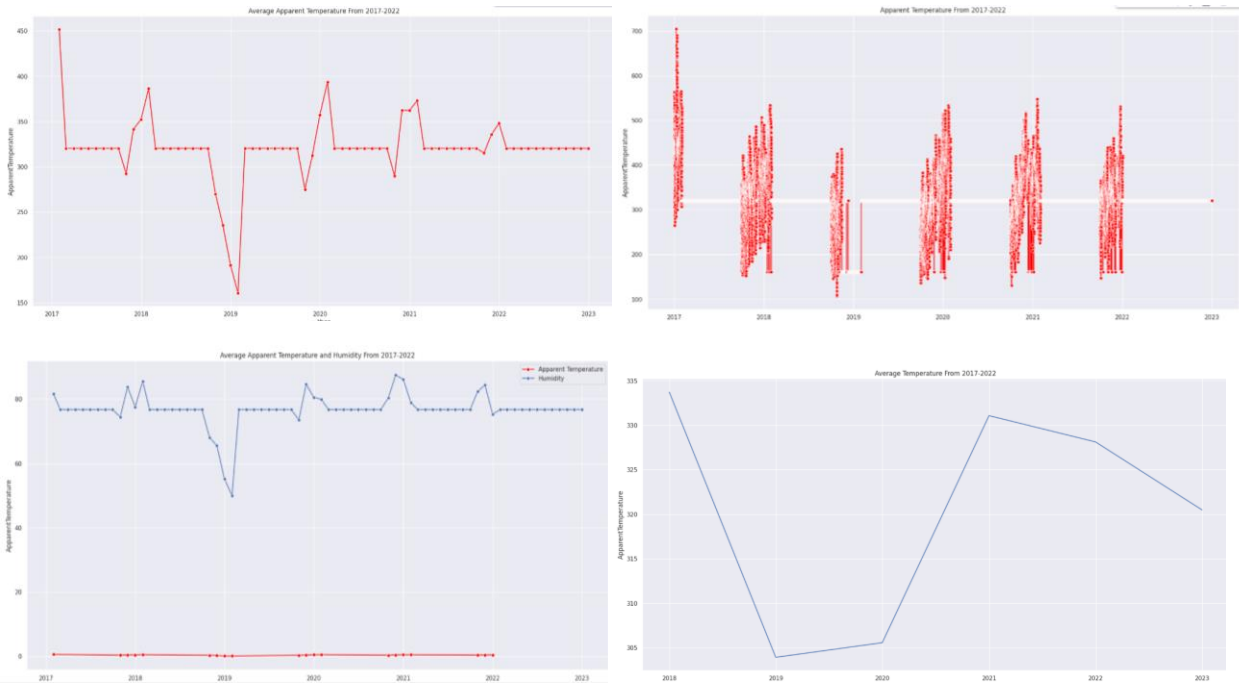
Vizualizimi i të dhënave është një komponent kritik në analizën e të dhënave, duke ofruar një mjet efektiv për të kuptuar, identifikuar modelet, dhe për të paraqitur informacionin. Këtu janë disa arsye se pse vizualizimi i të dhënave është i rëndësishëm:

- ***Kuptimi i të dhënave:*** Vizualizimi bën të dhënat të shihen më lehtë dhe të kuptohen më mirë nga përdoruesit. Një grafik i mirë mund të ndihmojë në identifikimin e trendeve, strukturave, dhe anomalive në të dhënat.
- ***Identifikimi i modeleve dhe marrëdhënieve:*** Përmes vizualizimit, mund të identifikohen modele dhe marrëdhëniet midis të dhënave. Grafikët, siç janë linjat kohore, mund të tregojnë trende, korrelacione dhe ndërveprime midis variablave.
- ***Kumtesa e informacionit:*** Vizualizimi mund të paraqesë një sasi të madhe të informacionit në një formë të përdorshme. Një vizualizim i mirë mund të përmbajë shumë informacione në një formë të kompaktuar dhe të kuptueshme.
- ***Përcaktimi i strategjive dhe vendimmarrja e informuar:*** Bazuar në vizualizimet, vendimmarrësit mund të hartojnë strategji dhe të marrin vendime të informuara. Grafikët mund të ndihmojnë në përcaktimin e trendeve të mundshme dhe ndërveprimeve që mund të ndikojnë në marrjen e vendimeve.

```
# Visualizing data

# Apparent Temperature From 2017-2022
plt.figure(figsize=(20, 10))
plt.title("Apparent Temperature From 2017-2022")
sns.lineplot(x=df['datetime'], y=df['ApparentTemperature'], marker='o',
color='red')
plt.xlabel("Year")

# Average Temperature From 2017-2022
plt.figure(figsize=(20, 10))
plt.title('Average Apparent Temperature From 2017-2022')
sns.lineplot(x=monthly_data['datetime'],
y=monthly_data['ApparentTemperature'], marker='o', color='red')
plt.xlabel("Year")
```

Për sa i përket kodit në vazhdim, ky është një shembull i disa vizualizimeve të ndryshme duke përdorur bibliotekën Seaborn dhe Matplotlib në Python. Le të shpjegojmë disa nga veprimet kryesore në këtë kod:

- ***Vizualizimi i Apparent Temperature dhe lagështisë në periudhën 2017-2022:*** Përdoret një grafik i linjës për të shfaqur Apparent Temperature dhe lagështinë në periudhën kohore 2017-2018.
- ***Vizualizimi i temperaturës mesatare dhe lagështisë për muajt nga 2017-2022:*** Përdoret një tjetër grafik i linjës për të shfaqur temperaturën mesatare dhe lagështinë për muajt nga 2017-2022.



4.1. Vizualizimi i korrelacionit midis të dhënave:

Kodimi për korrelacionin kemi përdorur një heatmap për të shfaqur matricën e korrelacionit midis të gjitha kolonave, përveç kolonës së datës ('datetime'). Korrelacioni është një masë e ndërveprimit lineare midis dy variablave, e shprehur si vlera nga -1 deri në 1. Në heatmap, vlerat afër 1 tregojnë një korrelacion pozitiv të fortë, ato afër -1 tregojnë një korrelacion negativ të fortë, dhe vlerat afër 0 tregojnë një mungesë të korrelacionit. Le të shpjegojmë disa prej rezultateve të mundshme nga heatmap:

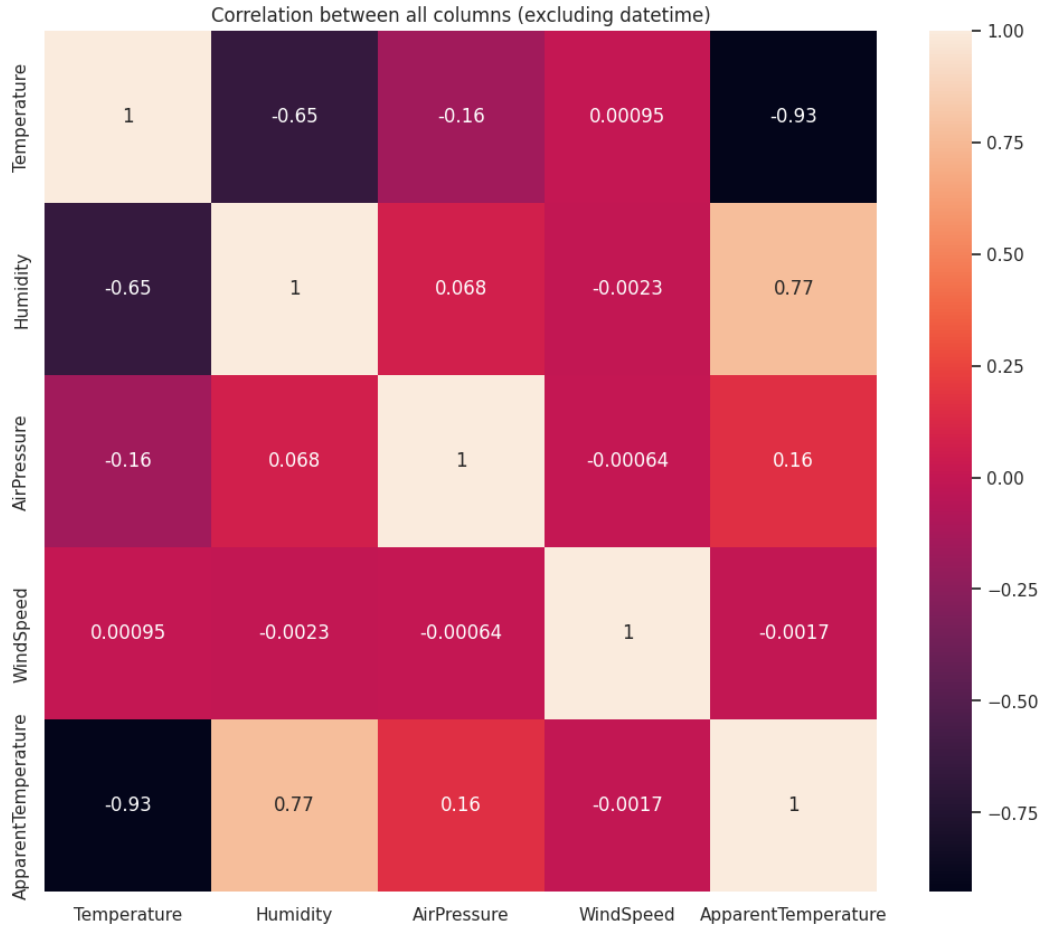
- ***Temperatura dhe Lagështia:*** Vlera e korrelacioni është (-0.65) midis temperaturës dhe lagështisë tregon për një korrelacion negativ të moderuar midis temperaturës dhe lagështisë. Kjo do të thotë se kur temperatura rritet, lagështia zakonisht bie, dhe anasjelltas.
- ***Lagështia dhe Shpejtësia e Erës:*** Vlera e korrelacionit mes lagështisë dhe shpejtësisë së erës është vlera (-0.0023). Korrelacioni afër zero tregon për mungesën e një marrëdhënie të dukshme lineare midis lagështisë dhe shpejtësisë së erës. Kjo do të thotë se ndryshimet në njërën variabël nuk lidhen në mënyrë të drejtpërdrejtë me ndryshimet në tjetrën.
- ***Lagështia dhe Presioni i Ajrit:*** Vlera e korrelacionit midis tyre është shumë s vogël (0.068) që tregon për një mungesë të lidhjes lineare midis lagështisë dhe presionit të ajrit. Në këtë rast, nuk ka një marrëdhënie lineare të dukshme midis këtyre dy variablave.
- ***Vlera e korrelacionit midis lagështisë dhe temperaturës ndjesuese (Apparent Temperature)*** është 0.77. Një vlerë e korrelacionit prej 0.77 tregon për një korrelacion pozitiv të fortë midis lagështisë dhe temperaturës ndjesuese. Kjo do të thotë se kur lagështia rritet, tendenca është që edhe temperatura ndjesuese të rritet. Vlera pozitive e korrelacionit tregon për një marrëdhënie lineare pozitive midis këtyre dy variablave.

- ***Vlera e korrelacionit midis temperaturës dhe temperaturës ndjesuese (Apparent Temperature)*** është -0.93. Një vlerë e korrelacionit prej -0.93 tregon për një korrelacion negativ shumë të fortë midis temperaturës dhe temperaturës ndjesuese. Kjo do të thotë se kur temperatura rritet, tendenca është që temperatura ndjesuese të bie. Vlera negative e korrelacionit tregon për një marrëdhënie lineare negative midis këtyre dy variablave.

Përgjithësisht, korrelacioni është një mjet i rëndësishëm për të kuptuar marrëdhëniet midis të dhënave, por nuk tregon shkak-efekt. Është e rëndësishme interpretuar këto rezultate duke i konsideruar rrethanat dhe njohuritë e domosdoshme për fushën e studiuar.

```
plt.figure(figsize=(12, 10))
plt.title("Correlation between all columns (excluding datetime)")
sns.heatmap(data=df.drop('datetime', axis=1).corr(), cmap="rocket",
annot=True)
plt.show()
```

OUTPUT:



4.2. Paraqitja e Regresionit ndërmjet Lagështisë dhe Apperent Temperature

Regresioni linear është një mënyrë e modelimit statistik për të studiuar marrëdhënien lineare mes dy ose më shumë variablave. Në një regresion linear, synohet të gjejmë një linjë tërthore (lineare) që përcakton marrëdhënien midis variablave. Kjo linjë mund të përshkruajë trendin e të dhënave ose të parashikojë vlerat e reja bazuar në të dhënat ekzistuese.

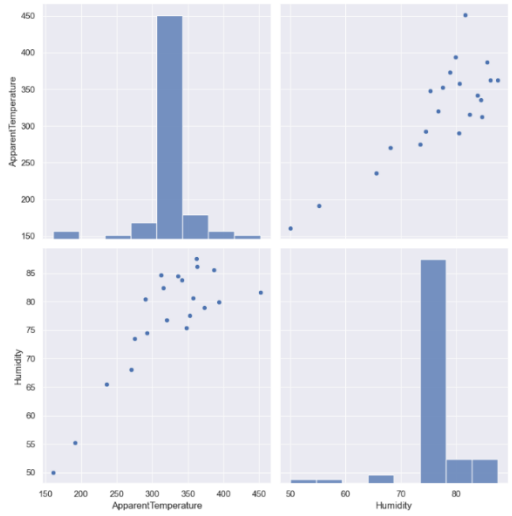
Në përdorimin e regresionit linear, kemi një variabël pavarëse (variabla shpjeguese) dhe një variabël të pavarur (variabla përkujdesëse). Modeli ynë do të tentojë të parashikojë variabël pavarëse bazuar në vlerat e variablave përkujdesëse.

Në figura, një regresion linear mund të dallohet nga një linjë e drejtë që përpiqet të përputhet me të dhënat. Nëse të dhënat janë të shpërndara rreth kësaj linje me një model linear, kjo është një tregues i mirë se regresioni linear është një model i përshtatshëm për të dhënat.

Një aspekt tjetër i vlerësimit të një modeli të regresionit linear është analiza e koeficientëve të modelit dhe interpretimi i tyre. Koeficientët tregojnë ndryshimin e pritur në variabël pavarëse për një ndryshim të njësi në variabël përkujdesëse, dhe kjo është një pjesë e rëndësishme e kuptimit të ndikimit të variablave mbi njëra-tjetrën.

Sqarim: Për shkakë të problemeve për instalimin e sklearn në PyCharm dhe poashtu në Google Collab nuk kemi mundur të paraqesim nëse kjo paraqet Regresion Linear apo JoLinear po në bazë të figures poshtë mendojmë që paraqet Regresion Linear.

```
AT_H = monthly_data[['ApparentTemperature', 'Humidity']]
print(AT_H)

sns.pairplot(AT_H, kind='scatter').fig.set_size_inches(10, 10)
```



Pra siç mund të vërehet edhe nga figura rezltatet e Apparent Temperature dhe Humidity po shkojnë në të njëtën vijë që edhe mendohet se paraqet Regresion Linear.

4.2 Vizualizime përmbyllëse të pjesës së pare

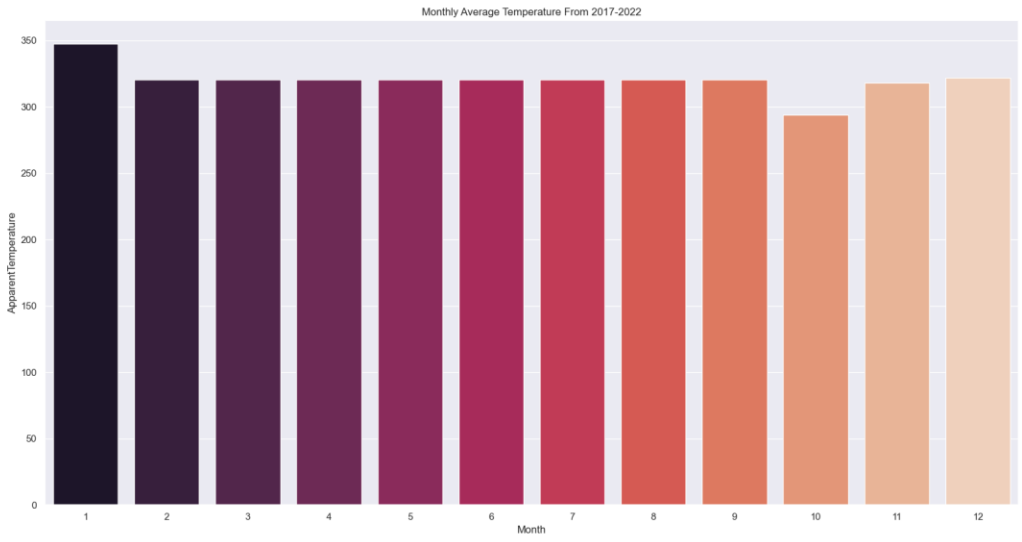
Kodi që kemi prezantar bën disa vizualizime të dhënash nga dataseti , duke përdorur bibliotekat `matplotlib` dhe `seaborn` në Python. Le të shpjegojmë secilin pjesë të kodit:

- ***Grafiku i Mesatares Mujore të Temperaturës nga 2017-2022:***

```
   # Monthly Average Temperature From 2017-2022
plt.figure(figsize=(20, 10))
plt.title("Monthly Average Temperature From 2017-2022")
sns.barplot(x='datetime', y='ApparentTemperature', data=avg_monthly_temp,
palette="rocket")
plt.xlabel("Month")
plt.show()
```

Ky fragment krijon një grafik shpërthyese (bar plot) që paraqet mesataren mujore të temperaturës ndjesuese nga viti 2017 deri në 2022. Përdorimi i `seaborn.barplot()` shërben për të shfaqur mesataren e temperaturës për çdo muaj në një mënyrë vizuale. Ngjyrat janë zgjedhur përmes paletës "rocket".

OUTPUT:

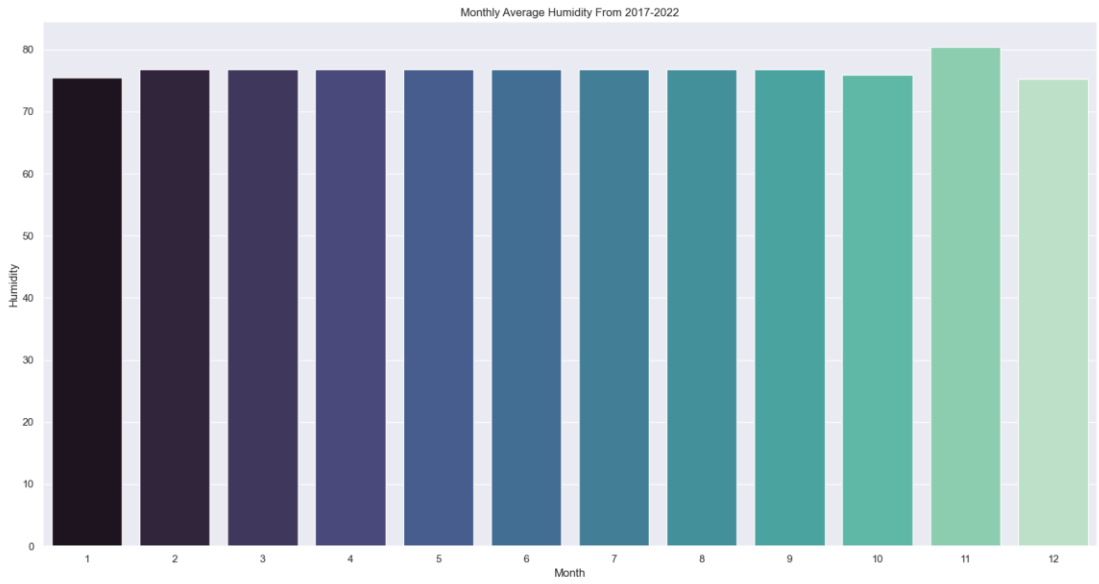


- ***Grafiku i Mesatares Mujore të Lageshtisë nga 2017-2022:***

```
   #Monthly Average Humidity From 2017-2022
plt.figure(figsize=(20, 10))
plt.title("Monthly Average Humidity From 2017-2022")
sns.barplot(x=avg_monthly_humidity['datetime'],
y=avg_monthly_humidity['Humidity'], palette="mako")
plt.xlabel("Month");
plt.show()
```

Kjo pjesë e kodit krijon një grafik shpërthyese që paraqet mesataren mujore të lagështisë nga viti 2017 deri në 2022. Përdorimi i `seaborn.barplot()` përdoret për të shfaqur mesataren e lagështisë për çdo muaj.

OUTPUT:

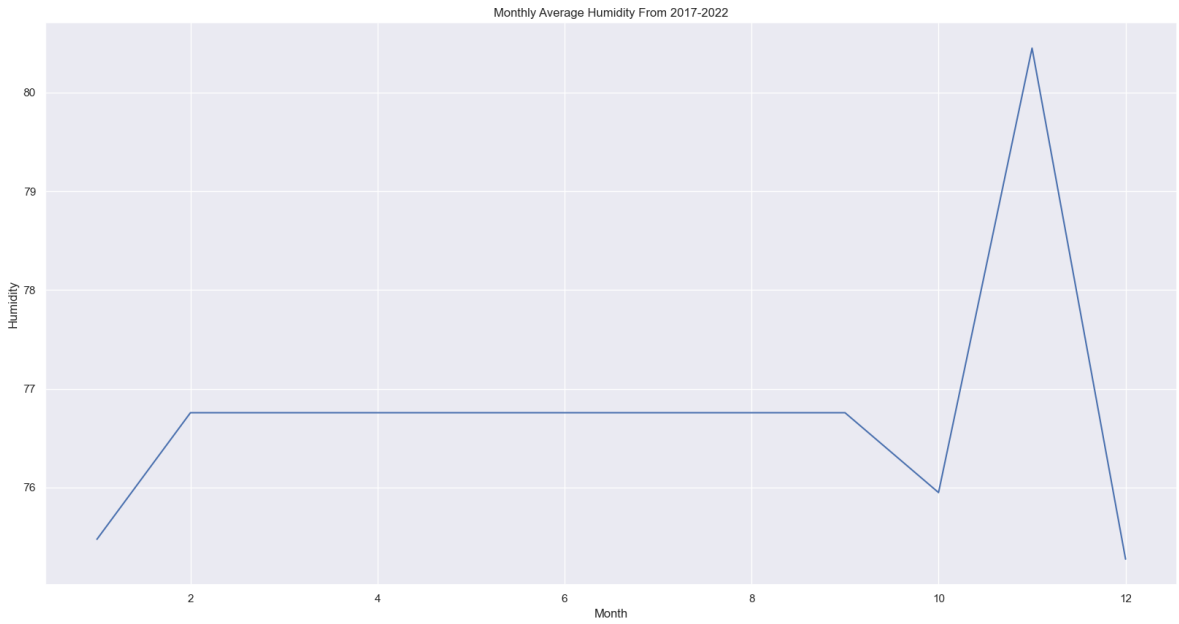


- ***Grafiku i Mesatajes Mujore të Lageshtisë nga 2017-2022 (Linjë):***

```
# Monthly Average Humidity From 2017-2022
plt.figure(figsize=(20, 10))
plt.title("Monthly Average Humidity From 2017-2022")
sns.lineplot(x='datetime', y='Humidity', data=avg_monthly_humidity)
plt.xlabel("Month")
plt.show()
```

Kjo pjesë e kodit krijon një grafik vijash (line plot) që paraqet mesataren mujore të lagështisë nga viti 2017 deri në 2022. Përdorimi i `seaborn.lineplot()` shërben për të shfaqur trendin e lagështisë për çdo muaj.

OUTPUT:

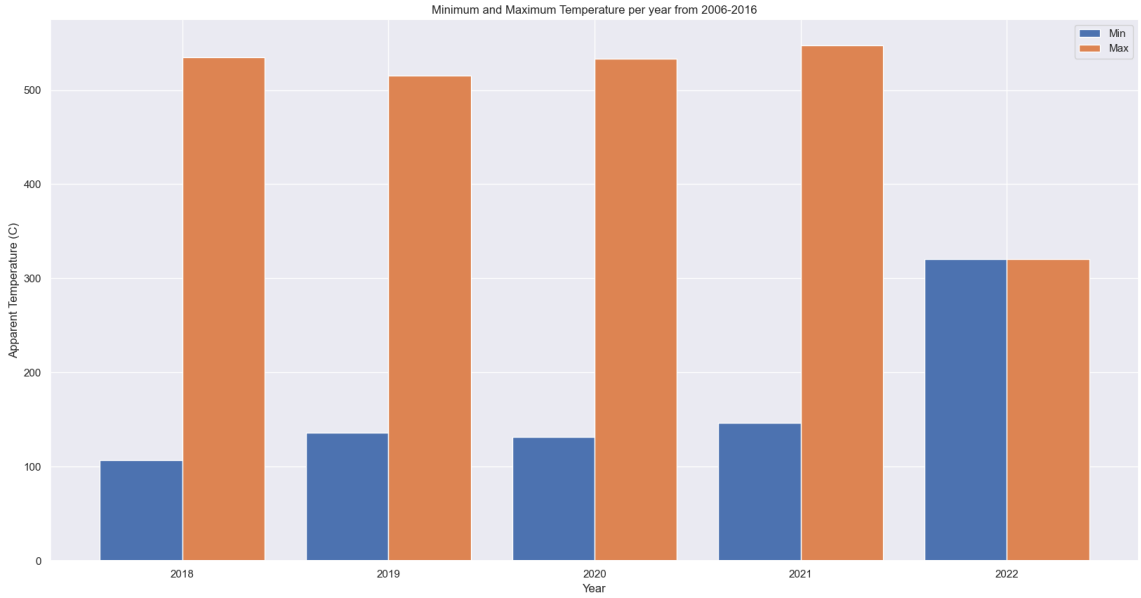


- ***Grafiku i Temperaturës së Maksimale dhe Minimale për Secilin Vit nga 2017-2022:***

```
   #Minimum and Maximum Temperature per year from 2017-2022
X_axis = np.arange(len(min_max_temp['datetime']))

plt.figure(figsize=(20, 10))

plt.bar(X_axis - 0.2, min_max_temp['Min Apparent Temperature (C)'], 0.4,
label='Min')
plt.bar(X_axis + 0.2, min_max_temp['Max Apparent Temperature (C)'], 0.4,
label='Max')

plt.xticks(X_axis, min_max_temp['datetime'])
plt.xlabel("Year")
plt.ylabel("Apparent Temperature (C)")
plt.title("Minimum and Maximum Temperature per year from 2006-2016")
plt.legend()
plt.show()
```

Kjo pjesë e kodit krijon një grafik shpërthyese që paraqet temperaturën maksimale dhe minimale për çdo vit nga 2017 deri në 2022. Përdorimi i `plt.bar()` përdoret për të shfaqur vlerat e minimumit dhe maksimumit si dy shtylla të ndara për secilin vit. Ky grafik shërben për të shfaqur ndryshimet e temperaturës ndjesuese në vitet e dhëna.

OUTPUT:


Minimum and Maximum Temperature per year from 2006-2016
Min
Max
Apparent Temperature (C)
0
100
200
300
400
500
2018
2019
2020
2021
2022
Year

5. *Vizualizimi i rezultateve të parashikimeve SARIMA*

Në periudhën kohore nga viti 2017 deri në vitin 2022, kemi pasur në dispozicion datasete të detajuara për temperaturën, lagështinë, presionin e ajrit dhe shpejtësinë e erës. Me qëllim të përmirësimit të analizës dhe për të paraqitur rezultate parashikuese më të avancuara, kemi shtuar një kolonë shtesë në secilin nga këto datasete. Kjo kolonë përmban rezultatet e parashikimit të bëra përmes një algoritmi të specializuar SARIMA (Seasonal Autoregressive Integrated Moving Average).

Analiza e parashikimit me algoritmin SARIMA është realizuar për të identifikuar modelet e sezonave, përfshirë përdorimin e vlerave të mëparshme për të parashikuar vlerat e ardhshme. Rezultatet e këtij procesi janë shtuar si një kolonë shtesë në çdo dataset përkatëse. Kjo është një përpjekje për të ofruar një pamje më të thelluar dhe më parashikuese të zhvillimeve të temperaturës, lagështisë, presionit të ajrit dhe shpejtësisë së erës në periudhën e studiuar.

Më poshtë, paraqesim grafikët për secilin dataset, duke shfaqur vlerat e vërtetuara dhe vlerat e parashikuara nga modeli SARIMA nëpërmjet një komponenti vizual. Ky hibrid i të dhënave të vërtetuara dhe atyre të parashikuara përbën një burim të fuqishëm për analizat dhe përmbledhjen e tendencave të mundshme në këto parametra atmosferike për periudhën e interesuar.

5.1. Vizualizimi i Rezultateve për Datasetin Temperatura

Ky kod është një shembull i analizës së të dhënave të temperaturave duke përdorur pandas, matplotlib dhe numpy në Python. Le të shpjegojmë secilën pjesë të kodit:

- *`import pandas as pd`:* Ky rresht importon librarinë pandas dhe e quan atë me shkurtimin `pd`. Pandas është një librari për manipulim dhe analizë të të dhënave.
- *`import matplotlib.pyplot as plt`:* Ky rresht importon modulin `pyplot` nga libraria Matplotlib dhe e quan atë me shkurtimin `plt`. Matplotlib është një librari për vizualizim të të dhënave.
- *`import numpy as np`:* Ky rresht importon librarinë numpy dhe e quan atë me shkurtimin `np`. Numpy është një librari për manipulim të matricave dhe ndërtime numërike.
- *`df = pd.read_csv(r'C:\Users\Admin\Desktop\Krahasimi_temperature.csv')`:* Ky rresht lexon një file CSV të dhënash të temperaturave dhe i ruajnë ato në një DataFrame (tabelë) të quajtur `df`.
- *`print("Column Names:", df.columns)`:* Ky rresht shfaq emrat e kolonave të DataFrame-it në terminal për të kontrolluar për gabime ose për sensitivitet të rastit.
- Për përcaktimin e formatit të datës në kolonat '2017' deri te '2022', përdoren funksionet *`pd.to_datetime`*. Pastaj, janë eleminuar rreshtat me vlera null në kolonat e datës.
- Pjesa e vizualizimit fillon me krijimin e një figure dhe dy nënvizime, njëra për temperaturat dhe tjetra për rezultatet e SARIMA.

- Pjesa e parë e vizualizimit shfaq temperaturat për secilin vit me vijat kohore. Për secilin vit, janë përdorur dy kolona për datën dhe temperaturat. Etiketat dhe titulli janë shtuar për të bërë vizualizimin më të lexueshëm.
- Pjesa e dytë e vizualizimit shfaq rezultatet e modelit SARIMA me vijat e sipërme dhe të poshtme të shqyrtimit, duke përdorur `fill_between` për të shfaqur një zonë ekuivale.
- *`plt.tight_layout()`:* Ky rresht përdoret për të siguruar që vizualizimi të jetë i shkëlqyeshëm pa ndërthurje të titujve apo etiketave.
- *`plt.show()`:* Ky rresht shfaq figurën e plotë me vizualizimin në dritaren e vizualizimit.

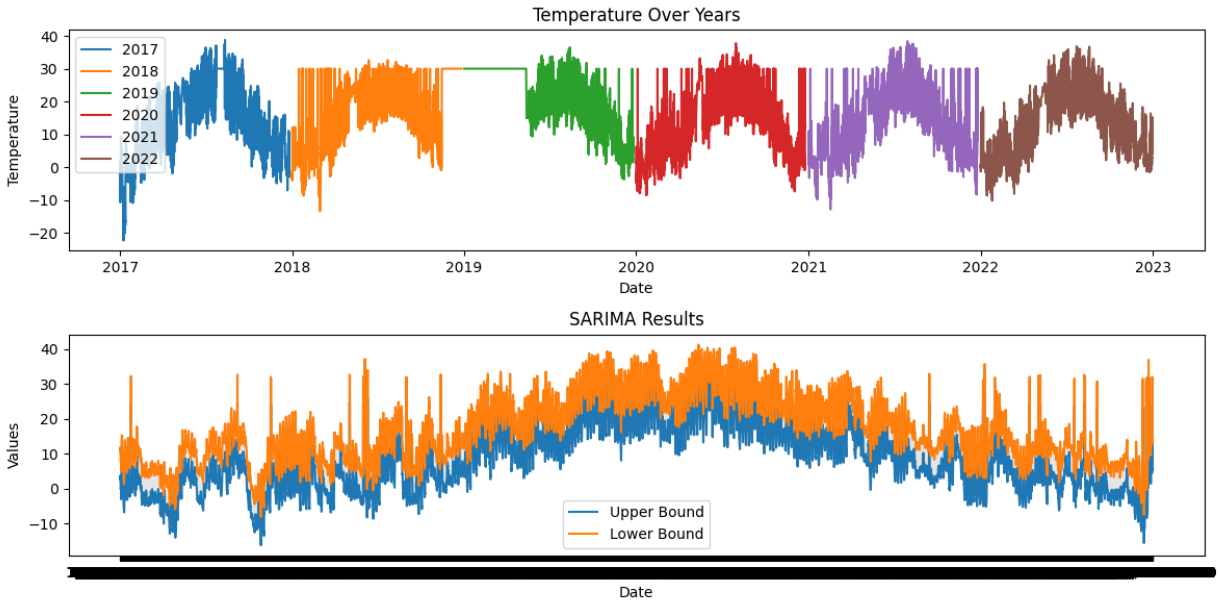
Ky kod është një shembull i analizës së të dhënave të temperaturave, duke përfshirë vizualizimin e të dhënave dhe rezultateve të modelit SARIMA.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Assuming your dataset is in a CSV file
df = pd.read_csv(r'C:\Users\Admin\Desktop\Krahasimi_temperature.csv')

# Print column names to check for typos or case sensitivity
print("Column Names:", df.columns)

# Combine datetime and temperature columns for each year
df['2017'] = pd.to_datetime(df['2017'], format='%m/%d/%Y %H:%M', utc=True,
errors='coerce')
df['2018'] = pd.to_datetime(df['2018'], format='%m/%d/%Y %H:%M', utc=True,
errors='coerce')
df['2019'] = pd.to_datetime(df['2019'], format='%m/%d/%Y %H:%M', utc=True,
errors='coerce')
df['2020'] = pd.to_datetime(df['2020'], format='%m/%d/%Y %H:%M', utc=True,
errors='coerce')
df['2021'] = pd.to_datetime(df['2021'], format='%m/%d/%Y %H:%M', utc=True,
errors='coerce')
df['2022'] = pd.to_datetime(df['2022'], format='%d.%m.%Y %H:%M:%S', utc=True,
errors='coerce')

# Drop rows with NaT values in datetime columns
df = df.dropna(subset=['2017', '2018', '2019', '2020', '2021', '2022'])

# Plot temperature data for each year
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
plt.plot(df['2017'], df['Unnamed: 1'], label='2017')
plt.plot(df['2018'], df['Unnamed: 3'], label='2018')
plt.plot(df['2019'], df['Unnamed: 5'], label='2019')
plt.plot(df['2020'], df['Unnamed: 7'], label='2020')
plt.plot(df['2021'], df['Unnamed: 9'], label='2021')
plt.plot(df['2022'], df['Unnamed: 11'], label='2022')

plt.title('Temperature Over Years')
plt.xlabel('Date')
```

```
plt.ylabel('Temperature')
plt.legend()

# Plot SARIMA results
plt.subplot(2, 1, 2)
plt.plot(df['SARIMA'], df['lower T_mu_actual'], label='Upper Bound')
plt.plot(df['SARIMA'], df['upper T_mu_actual'], label='Lower Bound')

# You can customize the SARIMA plot as needed
plt.fill_between(df['SARIMA'], df['lower T_mu_actual'], df['upper
T_mu_actual'], where=(~np.isnan(df['lower T_mu_actual']) &
~np.isnan(df['upper T_mu_actual'])), color='gray', alpha=0.2)

plt.title('SARIMA Results')
plt.xlabel('Date')
plt.ylabel('Values')
plt.legend()

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

OUTPUT:

```
Column Names: Index(['2017', 'Unnamed: 1', '2018', 'Unnamed: 3', '2019', 'Unnamed: 5',
       '2020', 'Unnamed: 7', '2021', 'Unnamed: 9', '2022', 'Unnamed: 11',
       'SARIMA', 'lower T_mu_actual', 'upper T_mu_actual'],
      dtype='object')
```



5.2. Vizualizimi i Rezultateve për Datasetin Humidity(Lageshtia)

Ky kod është një shembull i një analize të të dhënave lagështisë (humidity) duke përdorur pandas dhe matplotlib në Python. Le të shpjegojmë secilën pjesë të kodit:

- *`import pandas as pd`:* Ky rresht importon librarinë pandas dhe e quan atë me shkurtimin `pd`. Pandas është një librari për manipulim dhe analizë të të dhënave.
- *`import matplotlib.pyplot as plt`:* Ky rresht importon modulin `pyplot` nga libraria Matplotlib dhe e quan atë me shkurtimin `plt`. Matplotlib është një librari për vizualizim të të dhënave.
- *`from matplotlib.dates import DateFormatter`:* Ky rresht importon klasën *`DateFormatter`* nga moduli `dates` i Matplotlib, e cila mund të përdoret për formatimin e datave në vijat kohore.
- *`df = pd.read_csv(r'C:\Users\Admin\Desktop\humidity_krahasimi.csv')`*: Ky rresht lexon një file CSV të dhënash të ujit të ajrit dhe i ruajnë ato në një DataFrame (tabelë) të quajtur `df`.
- *`print("Column Names:", df.columns)`:* Ky rresht shfaq emrat e kolonave të DataFrame-it në terminal për të kontrolluar për gabime ose për sensitivitet të rastit.
- Për përcaktimin e formatit të datës në kolonat 'Unnamed', përdoren funksionet `pd.to_datetime`. Pastaj, radhiten kolonat sipas vitit dhe janë eleminuar rreshtat me vlera null në kolonat e datës.
- *`humidity_columns = ['Unnamed: 1', 'Unnamed: 3', 'Unnamed: 5', 'Unnamed: 7', 'Unnamed: 9', 'Unnamed: 11', 'Unnamed: 13', 'Unnamed: 14']`:* Ky rresht përcakton emrat e kolonave të lidhura me nivelet e ujit të ajrit.
- *`for col in humidity_columns: df[col] = df[col].astype(str)`:* Ky cikël konverton vlerat e kolonave të niveleve të ujit në string për t'i përdorur më pas në vizualizim.
- Pjesa e vizualizimit fillon me krijimin e një figure dhe dy nënvizime, njëra për nivelet e ujit dhe tjetra për rezultatet e SARIMA.
- Pjesa e parë e vizualizimit shfaq nivelet e lagështisë për secilin vit me vijat kohore. Për secilin vit, janë përdorur dy kolona për datën dhe nivelet e ujit. Etiketat dhe titulli janë shtuar për të bërë vizualizimin më të lexueshëm.
- Pjesa e dytë e vizualizimit shfaq rezultatet e modelit SARIMA me vijat e sipërme dhe të poshtme të shqyrtimit, duke përdorur `fill_between` për të shfaqur një zonë ekuivale
- *`plt.gca().xaxis.set_major_formatter(DateFormatter('%m/%d/%Y %H:%M'))`:* Ky rresht përdor klasën `DateFormatter` për të formatuar datat në vijën kohore të x-osis.
- *`plt.tight_layout()`:* Ky rresht përdoret për të siguruar që vizualizimi të jetë i shkëlqyeshëm pa ndërthurje të titujve apo etiketave.
- *`plt.show()`:* Ky rresht shfaq figurën e plotë me vizualizimin në dritaren e vizualizimit.

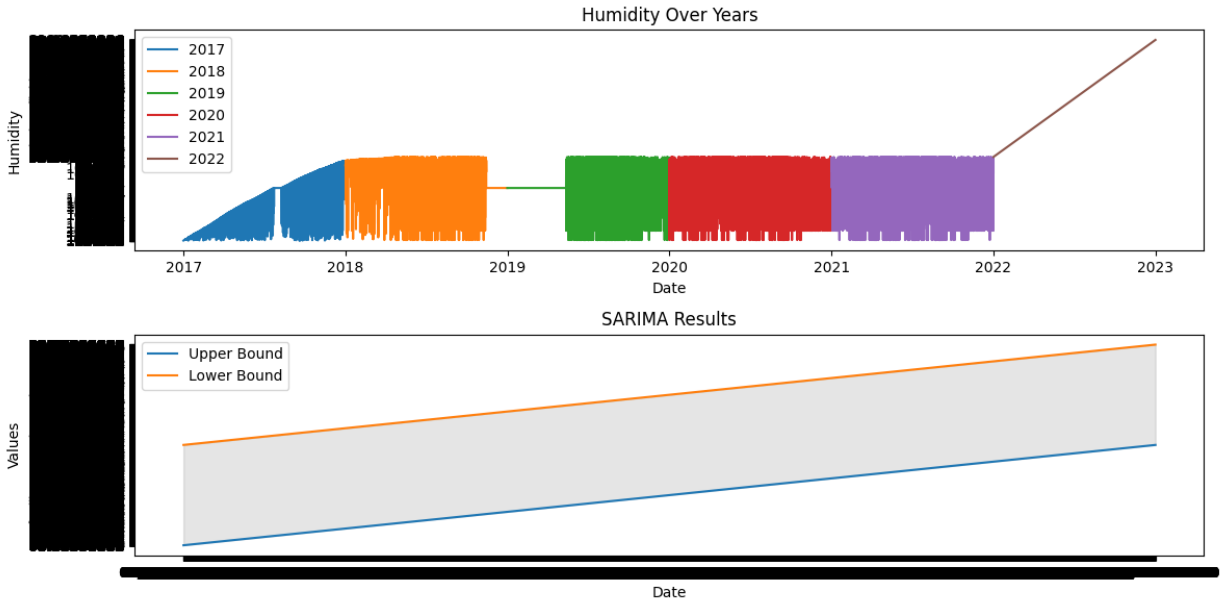
Ky kod është një shembull i analizës së të dhënave me përqendrim në nivelet lagështisë dhe përdorimin e modelit SARIMA për të parashikuar këto nivele.

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter

# Assuming your dataset is in a CSV file
df = pd.read_csv(r'C:\Users\Admin\Desktop\humidity_krahasimi.csv')

#Print column names to check for typos or case sensitivity
print("Column Names:", df.columns)

# Combine datetime and humidity columns for each year
df['Unnamed: 0'] = pd.to_datetime(df['Unnamed: 0'], format='%m/%d/%Y %H:%M',
utc=True, errors='coerce')
df['Unnamed: 2'] = pd.to_datetime(df['Unnamed: 2'], format='%m/%d/%Y %H:%M',
utc=True, errors='coerce')
df['Unnamed: 4'] = pd.to_datetime(df['Unnamed: 4'], format='%m/%d/%Y %H:%M',
utc=True, errors='coerce')
df['Unnamed: 6'] = pd.to_datetime(df['Unnamed: 6'], format='%m/%d/%Y %H:%M',
utc=True, errors='coerce')
df['Unnamed: 8'] = pd.to_datetime(df['Unnamed: 8'], format='%m/%d/%Y %H:%M',
utc=True, errors='coerce')
df['Unnamed: 10'] = pd.to_datetime(df['Unnamed: 10'], format='%m/%d/%Y
%H:%M', utc=True, errors='coerce')

# Drop rows with NaT values in datetime columns
df = df.dropna(subset=['Unnamed: 0', 'Unnamed: 2', 'Unnamed: 4', 'Unnamed:
6', 'Unnamed: 8', 'Unnamed: 10'])

# Convert y-values to strings
humidity_columns = ['Unnamed: 1', 'Unnamed: 3', 'Unnamed: 5', 'Unnamed: 7',
'Unnamed: 9', 'Unnamed: 11', 'Unnamed: 13', 'Unnamed: 14']

for col in humidity_columns:
    df[col] = df[col].astype(str)

# Plot humidity data for each year
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
plt.plot(df['Unnamed: 0'], df['Unnamed: 1'], label='2017')
plt.plot(df['Unnamed: 2'], df['Unnamed: 3'], label='2018')
plt.plot(df['Unnamed: 4'], df['Unnamed: 5'], label='2019')
plt.plot(df['Unnamed: 6'], df['Unnamed: 7'], label='2020')
plt.plot(df['Unnamed: 8'], df['Unnamed: 9'], label='2021')
plt.plot(df['Unnamed: 10'], df['Unnamed: 11'], label='2022')

plt.title('Humidity Over Years')
plt.xlabel('Date')
plt.ylabel('Humidity')
plt.legend()

# Plot SARIMA results
plt.subplot(2, 1, 2)
plt.plot(df['SARIMA'], df['Unnamed: 13'], label='Upper Bound')
plt.plot(df['SARIMA'], df['Unnamed: 14'], label='Lower Bound')

# You can customize the SARIMA plot as needed
```

```
plt.fill_between(df['SARIMA'], df['Unnamed: 13'], df['Unnamed: 14'],
color='gray', alpha=0.2)

# Format x-axis dates
plt.gca().xaxis.set_major_formatter(DateFormatter('%m/%d/%Y %H:%M'))

plt.title('SARIMA Results')
plt.xlabel('Date')
plt.ylabel('Values')
plt.legend()

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

OUTPUT:

```
Column Names: Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4',
       'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9',
       'Unnamed: 10', 'Unnamed: 11', 'SARIMA', 'Unnamed: 13', 'Unnamed: 14'],
      dtype='object')
```



5.3. Vizualizimi i Rezultateve për Datasetin WindSpeed(Shpejtesia e erës)

Kodi që kemi shënuar për vizualizimin e këtyre detajeve në Python kemi përdorur pjesë të paketave për të lexuar një file CSV duke përdorur pandas dhe pastaj vizualizuar disa kolona të përzgjedhura në grafika të ndara me matplotlib. Le të shpjegojmë kodin hap pas hapi:

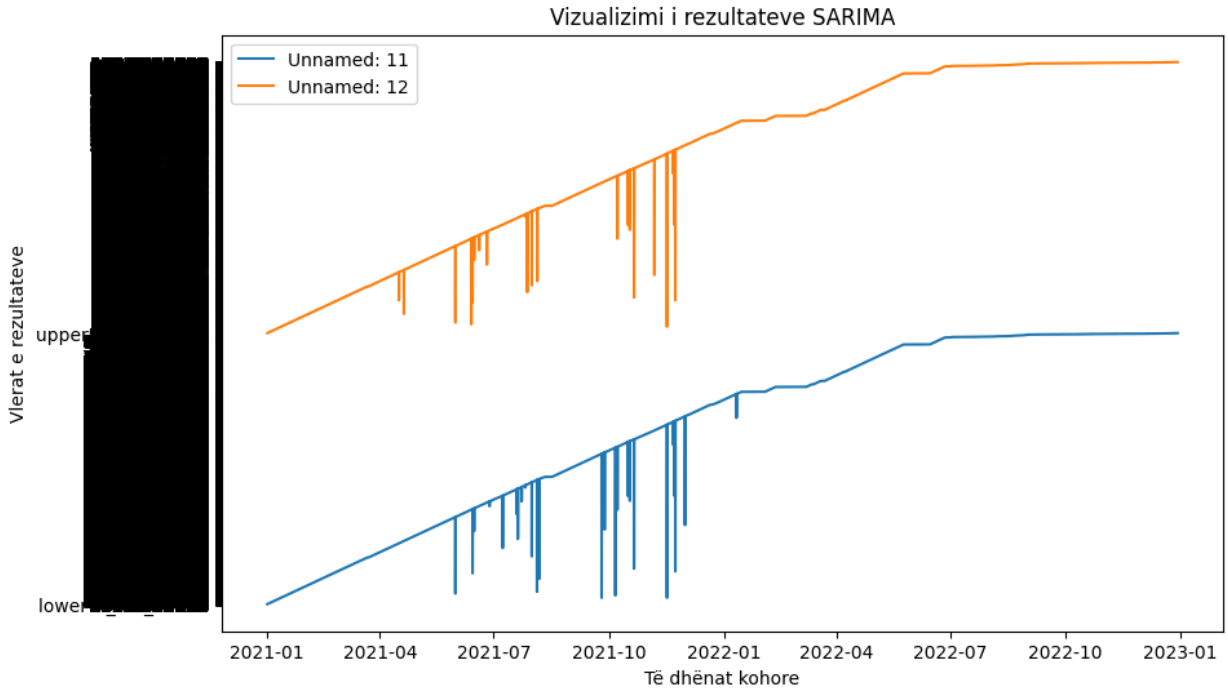
1. *`import pandas as pd`:* Këtu, ju importoni librarinë pandas dhe e shkurtuat emrin e saj në `pd`, që është praktikë e zakonshme.
2. *`import matplotlib.pyplot as plt`:* Ky import është për bibliotekën Matplotlib dhe e shkurtuat emrin e saj në `plt`, gjithashtu një praktikë e zakonshme.
3. *`df = pd.read_csv(r'/content/krahasimi_windspeed.csv')`:* Ky rresht lexon file-in CSV nga një rrugë e caktuar dhe e ruajnë në një DataFrame të pandas, i cili quhet `df`.
4. *`print("Kolonat e dataset-it:")`:* Ky rresht shton një shënim për të shfaqur në konsolë se cilat janë kolonat e dataset-it.
5. *`print(df.columns)`:* Ky rresht shfaq në konsolë emrat e kolonave të dataset-it duke përdorur atributin `columns` të DataFrame `df`.
6. *`df['SARIMA'] = pd.to_datetime(df['SARIMA'], format='%m/%d/%Y %H:%M', errors='coerce')`:* Ky rresht përdor funksionin `pd.to_datetime` për të konvertuar vlerat në kolonën 'SARIMA' në formën e datetime. Parametri `format` është përdorur për të specifikuar formatin e datës, ndërsa `errors='coerce'` përpunon gabimet duke zëvendësuar vlerat e pavlefshme me NaT (Not a Time).
7. *`selected_columns = df[['SARIMA', 'Unnamed: 11', 'Unnamed: 12']]`:* Ky rresht krijon një DataFrame të ri, i cili përfshin vetëm kolonat e përzgjedhura *('SARIMA', 'Unnamed: 11', 'Unnamed: 12')* nga DataFrame origjinal `df`.
8. *`selected_columns.set_index('SARIMA', inplace=True)`:* Ky rresht vendos kolonën 'SARIMA' si indeks të ri në DataFrame-in `selected_columns`. Parametri `inplace=True` do të thotë që ndryshimet të ndodhin direkt në objektin ekzistues, pa krijuar një kopje të re.
9. *`plt.figure(figsize=(10, 6))`*: Krijon një figurë me një dimension prej 10 në gjatësi dhe 6 në lartësi për të vendosur grafikat.
10. *`for column in selected_columns.columns: ...`:* Krijon një loop për secilën kolonë në DataFrame-in `selected_columns`.
11. *`plt.plot(selected_columns.index, selected_columns[column], label=column)`:* Krijon një grafik për secilën kolonë në indeksin e përzgjedhur, duke përdorur `plt.plot()`. Për secilën kolonë, shtohet një etiketë në legjendë me emrin e kolonës.
12. *`plt.title('Vizualizimi i rezultateve SARIMA')`:* Vendos një titull për grafikën.
13. *`plt.xlabel('Të dhënat kohore')`:* Vendos një etiketë për aksin X.
14. *`plt.ylabel('Vlerat e rezultateve')`:* Vendos një etiketë për aksin Y.
15. *`plt.legend()`:* Shfaq legjendën e grafikave.
16. *`plt.show()`:* Shfaq grafikën.

Në këtë kodë, po përdoret pandas për të manipuluar dhe analizuar të dhënat, ndërsa Matplotlib përdoret për të vizualizuar tokegrafikët.

```
import pandas as pd
import matplotlib.pyplot as plt

# Lexo file-in CSV duke përdorur pandas
df = pd.read_csv(r'/content/krahasimi_windspeed.csv')

# Shfaq kolonat e dataset-it
print("Kolonat e dataset-it:")
print(df.columns)

# Përcakto formatin e datës në kolonën 'SARIMA'
df['SARIMA'] = pd.to_datetime(df['SARIMA'], format='%m/%d/%Y %H:%M',
errors='coerce')

# Përzgjidh vetëm kolonat e interesuara
selected_columns = df[['SARIMA', 'Unnamed: 11', 'Unnamed: 12']]

# Vendos kolonën 'SARIMA' si indeks
selected_columns.set_index('SARIMA', inplace=True)

# Vizualizo rezultatet në grafika të ndara për secilën kolonë
plt.figure(figsize=(10, 6))
for column in selected_columns.columns:
    plt.plot(selected_columns.index, selected_columns[column],
label=column)

plt.title('Vizualizimi i rezultateve SARIMA')
plt.xlabel('Të dhënat kohore')
plt.ylabel('Vlerat e rezultateve')
plt.legend()
plt.show()
```

OUTPUT:

```
Kolonat e dataset-it:
Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4',
       'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9',
       'SARIMA', 'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13'],
      dtype='object')
```



5.4. *Vizualizimi i rezultateve për Datasetin AirPreasure(Presioni i Ajrit)*

Ky kod është një shembull i një analize të thjeshtë të të dhënave duke përdorur libraritë e Python si pandas dhe matplotlib. Le të shpjegojmë secilën pjesë të kodit:

- *`import pandas as pd`:* Ky rresht i kodit importon librarinë pandas dhe e quan atë me emrin shkurt `pd`. Pandas është një librar për analizë të të dhënave në Python.
- *`import matplotlib.pyplot as plt`:* Ky rresht i kodit importon modulin `pyplot` nga libraria Matplotlib dhe e quan atë me emrin shkurt `plt`. Matplotlib është një librar për vizualizim të të dhënave në Python.
- *`df = pd.read_csv(r'/content/krahasimi_airpreasure.csv')`:* Ky rresht lexon një file CSV duke përdorur pandas dhe e ruajnë në një DataFrame (tabelë) të quajtur `df`. Adresa e file-it është specifikuar në path-in e rreshtit.
- *`print("Kolonat e dataset-it:")`:* Ky rresht shfaq emrat e kolonave të DataFrame-it në terminal.
- *`df['SARIMA'] = pd.to_datetime(df['SARIMA'], format='%m/%d/%Y %H:%M', errors='coerce')`:* Ky rresht konverton kolonën 'SARIMA' në formatin e datetime duke përdorur pandas. `coerce` përdoret për të trajtuar çdo vlerë e cila nuk është e vlefshme si datetime si një vlerë null.

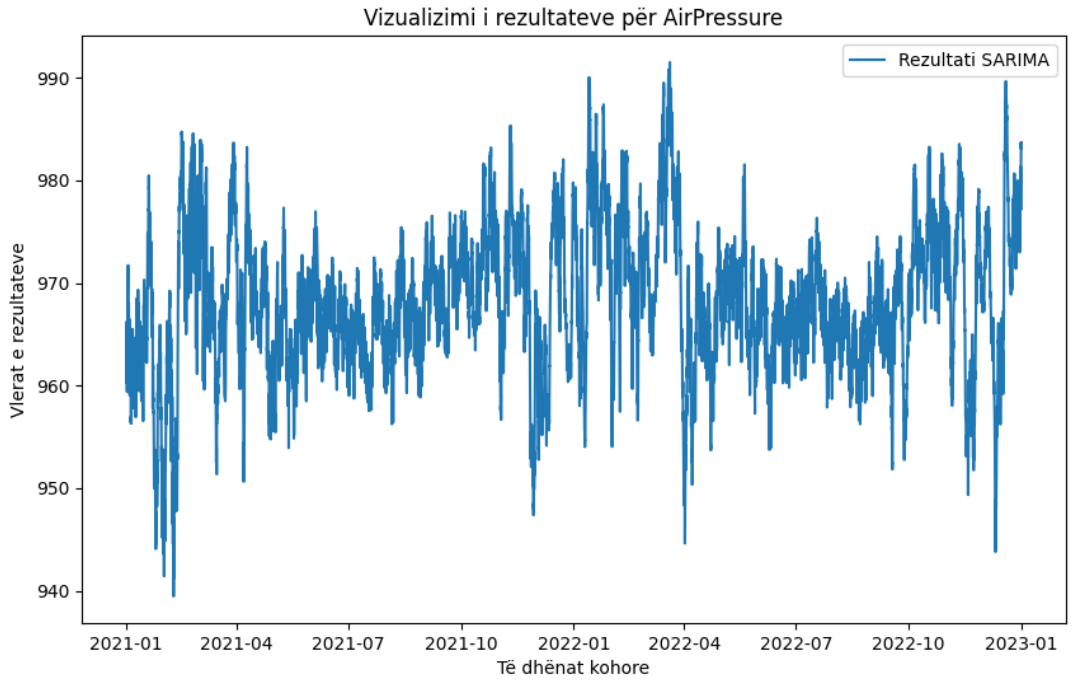
- *`selected_columns_airpressure = df[['SARIMA', 'Unnamed: 12']]`:* Ky rresht krijon një DataFrame të ri (`selected_columns_airpressure`) duke përzgjedhur vetëm dy kolona, 'SARIMA' dhe 'Unnamed: 12', nga DataFrame-i origjinal.
- *`selected_columns_airpressure.set_index('SARIMA', inplace=True)`:* Ky rresht vendos kolonën 'SARIMA' si indeks të DataFrame-it të përzgjedhur.
- *`selected_columns_airpressure.rename(columns={'Unnamed: 12': 'Rezultati SARIMA'}, inplace=True)`:* Ky rresht riemëron kolonën 'Unnamed: 12' në 'Rezultati SARIMA'.
- `plt.figure(figsize=(10, 6))`: Ky rresht krijon një figurë për vizualizimin me një madhësi të caktuar (10 piksela në gjatësi, 6 në lartësi)
- *`for column in selected_columns_airpressure.columns: plt.plot (selected_columns _airpressure.index, selected_columns_airpressure[column], label=column)`:* Ky rresht përdor një cikël për të vizualizuar secilën kolonë të përzgjedhur në grafik. `plot` është një funksion i matplotlib për të vizualizuar seria kohore.
- *`plt.title('Vizualizimi i rezultateve për AirPressure')`:* Ky rresht vendos titullin e grafikut.
- *`plt.xlabel('Të dhënat kohore')`* dhe *`plt.ylabel('Vlerat e rezultateve')`:* Këto rreshta vendosin etiketat për boshtët x dhe y të grafikut.
- *`plt.legend()`:* Ky rresht shton një legjendë në grafik për të shpjeguar çdo linjë në grafik.
- *`plt.show()`:* Ky rresht shfaq grafikun në dritaren e vizualizimit.

```
import pandas as pd
import matplotlib.pyplot as plt

# Lexo file-in CSV duke përdorur pandas
df = pd.read_csv(r'C:\Users\Admin\Desktop\krahasimi_airpreasure.csv')

# Shfaq kolonat e dataset-it
print("Kolonat e dataset-it:")
print(df.columns)

# Përcakto formatin e datës në kolonën 'SARIMA'
df['SARIMA'] = pd.to_datetime(df['SARIMA'], format='%m/%d/%Y %H:%M',
errors='coerce')

# Përzgjidh vetëm kolonat e interesuara për 'AirPressure'
selected_columns_airpressure = df[['SARIMA', 'Unnamed: 12']]

# Vendos kolonën 'SARIMA' si indeks
selected_columns_airpressure.set_index('SARIMA', inplace=True)

# Riemëro kolonën 'Unnamed: 12' si 'Rezultati SARIMA'
selected_columns_airpressure.rename(columns={'Unnamed: 12': 'Rezultati
SARIMA'}, inplace=True)

# Vizualizo rezultatet për 'AirPressure' me kolonën e riemëruar
plt.figure(figsize=(10, 6))
for column in selected_columns_airpressure.columns:
    plt.plot(selected_columns_airpressure.index,
selected_columns_airpressure[column], label=column)
```

```
plt.title('Vizualizimi i rezultateve për AirPressure')
plt.xlabel('Të dhënat kohore')
plt.ylabel('Vlerat e rezultateve')
plt.legend()
plt.show()
```

OUTPUT:

```
Kolonat e dataset-it:
Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4',
       'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9',
       'Unnamed: 10', 'SARIMA', 'Unnamed: 12'],
```



6.Konkluzioni

Ky projekt hulumtues ka shënuar një përpjekje të rëndësishme për të kuptuar dhe analizuar dinamikat e kushteve meteorologjike në Kosovë nga viti 2017 deri në vitin 2022. Qëllimi kryesor ka qenë të identifikohen lidhjet dhe ndërveprimet midis temperaturës, lagështisë, shpejtësisë së erës dhe presionit të ajrit në këtë periudhë kohore.

Në detyrën e parë, analiza e korrelacionit dhe regresionit linear ka përmirësuar kuptimin tonë për marrëdhëniet komplekse midis këtyre parametrave meteorologjikë. Përdorimi i gjuhës programuese Python për këtë analizë ka sjellë një avancim të procesit analitik dhe ka mundësuar identifikimin e marrëdhënieve të rëndësishme.

Detyra e dytë e projektit ka përfshirë përdorimin e algoritmit SARIMA për të parashikuar zhvillimet e mundshme të temperaturës, lagështisë, shpejtësisë së erës dhe presionit të ajrit. Përmes vizualizimeve grafike të përpunuara, rezultatet e parashikimeve janë prezantuar në një mënyrë të qartë dhe të kuptueshme. Kjo pjesë e projektit ka shtuar një dimension shtesë në interpretimin e tendencave dhe variacioneve të këtyre parametrave.

Në përgjithësi, ky projekt ka dhënë një mundësi të shkëlqyer në zhvillimin e aftësive tona për zhvillim të mëtejshëm të analizës së të dhënave si dhe njoftimin ma statistika të rëndësishme në fushën e analizës meteorologjike në Kosovë, duke përdorur një kombinim të analizave statistikore dhe algoritmave të avancuara për parashikim. Rezultatet e këtij projekti ofrojnë një bazë të mirë për kuptimin e dinamikave të kushteve atmosferike, duke sjellë vlerë për hartimin e politikave dhe vendimeve të bazuara në dije për menaxhimin e kushteve meteorologjike në vend. Në veçanti, fakti që janë përfunduar analiza të tilla si analiza e korrelacionit dhe përdorimi i algoritmave për parashikim tregon një qasje të pasur dhe tërësore në analizën e të dhënave meteorologjike.

Referencat

- Aditya Rambhad, Jan 2,2022, Performing Analysis of Meteorigical Data Using Python, https://medium.com/@adityasrambhad/performing-analysis-of-meteorological-data-using-python-fbeabc1cf798.
- Abhinandan Katoch, Mar 8,2022, Data Analytics using Python. https://medium.com/@abhinandankatoch/data-analytics-using-python-a801592f2fa6
- OpenAI. (2022, January 28). ChatGPT. Retrieved from https://www.openai.com/chatgp