Home-Assignment

October 16, 2023

[36]:
```
!pip install gdown
```

```
Requirement already satisfied: gdown in c:\users\malir\anaconda3\lib\site-
packages (4.7.1)
Requirement already satisfied: requests[socks] in
c:\users\malir\anaconda3\lib\site-packages (from gdown) (2.28.1)
Requirement already satisfied: beautifulsoup4 in
c:\users\malir\anaconda3\lib\site-packages (from gdown) (4.11.1)
Requirement already satisfied: filelock in c:\users\malir\anaconda3\lib\site-
packages (from gdown) (3.6.0)
Requirement already satisfied: six in c:\users\malir\anaconda3\lib\site-packages
(from gdown) (1.16.0)
Requirement already satisfied: tqdm in c:\users\malir\anaconda3\lib\site-
packages (from gdown) (4.64.1)
Requirement already satisfied: soupsieve>1.2 in
c:\users\malir\anaconda3\lib\site-packages (from beautifulsoup4->gdown) (2.3.1)
Requirement already satisfied: charset-normalizer<3,>=2 in
c:\users\malir\anaconda3\lib\site-packages (from requests[socks]->gdown) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\malir\anaconda3\lib\site-packages (from requests[socks]->gdown) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\malir\anaconda3\lib\site-packages (from requests[socks]->gdown)
(2022.9.14)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\malir\anaconda3\lib\site-packages (from requests[socks]->gdown)
(1.26.11)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
c:\users\malir\anaconda3\lib\site-packages (from requests[socks]->gdown) (1.7.1)
Requirement already satisfied: colorama in c:\users\malir\anaconda3\lib\site-
packages (from tqdm->gdown) (0.4.5)
```

[37]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
import gdown
import zipfile
```

[38]:
```
warnings.filterwarnings('ignore')

file_id = '1QnpOWeRAlycJLGsr4ghyB-TzS4cQpaLl'
url = f'https://drive.google.com/uc?id={file_id}'
output = 'file.zip'

gdown.download(url, output, quiet=False)

extracted = "HomeAssignment"
with zipfile.ZipFile('file.zip', 'r') as zip_ref:
    zip_ref.extractall(extracted)

csv_file_path = f"{extracted}/
 ↪Conditions_Contributing_to_COVID-19_Deaths__by_State_and_Age__Provisional_2020-2023.
 ↪csv"
df = pd.read_csv(csv_file_path)
df.head(10)
```

```
Downloading…
From: https://drive.google.com/uc?id=1QnpOWeRAlycJLGsr4ghyB-TzS4cQpaLl
To: C:\Users\malir\file.zip
100%|
 | 3.66M/3.66M [00:00<00:00, 6.18MB/s]
```

[38]:

	Data As Of	Start Date	End Date	Group	Year	Month	State	Condition Group	Condition	ICD10_codes	Age Group
0	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	0-24
1	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	25-34
2	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	35-44
3	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	45-54
4	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	55-64
5	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	65-74
6	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	75-84
7	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	85+
8	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	Not stated
9	06/25/2023	01/01/2020	06/24/2023	By Total	NaN	NaN	United States	Respiratory diseases	Influenza and pneumonia	J09-J18	All Ages

	COVID-19 Deaths	Number of Mentions	Flag
0	1554.0	1630.0	NaN
1	5775.0	5998.0	NaN
2	15026.0	15643.0	NaN
3	37335.0	38794.0	NaN
4	82382.0	85404.0	NaN
5	128349.0	132400.0	NaN
6	137362.0	140693.0	NaN
7	119833.0	121695.0	NaN
8	12.0	12.0	NaN
9	527628.0	542269.0	NaN

[39]:
```
df.describe()
```

[39]:

	Year	Month	COVID-19 Deaths	Number of Mentions
count	571320.000000	521640.000000	4.111710e+05	4.167610e+05
mean	2021.304348	6.071429	1.266096e+02	1.362334e+02
std	1.039850	3.425349	3.052289e+03	3.279466e+03
min	2020.000000	1.000000	0.000000e+00	0.000000e+00
25%	2020.000000	3.000000	0.000000e+00	0.000000e+00
50%	2021.000000	6.000000	0.000000e+00	0.000000e+00
75%	2022.000000	9.000000	1.900000e+01	2.100000e+01
max	2023.000000	12.000000	1.135624e+06	1.135624e+06

[40]:
```
df["Start Date"] = pd.to_datetime(df["Start Date"], format='%m/%d/%Y')
df["End Date"] = pd.to_datetime(df["End Date"], format='%m/%d/%Y')

start_date_filtered = pd.to_datetime('01/01/2020', format='%m/%d/%Y')
end_date_filtered = pd.to_datetime('06/24/2023', format='%m/%d/%Y')

filtered_df = df[(df["Start Date"] == start_date_filtered) & (df["End Date"] ==␣
 ↪end_date_filtered)]
```

[41]:
```
filtered_df["Start Date"].unique()
```

[41]:
```
array(['2020-01-01T00:00:00.000000000'], dtype='datetime64[ns]')
```

[42]:
```
filtered_df["End Date"].unique()
```

[42]:
```
array(['2023-06-24T00:00:00.000000000'], dtype='datetime64[ns]')
```

[43]:
```
filtered_df["Age Group"].unique()
```

[43]:
```
array(['0-24', '25-34', '35-44', '45-54', '55-64', '65-74', '75-84',
       '85+', 'Not stated', 'All Ages'], dtype=object)
```

[44]:
```
for i in ["Not stated","All Ages"]:
  filtered_df = filtered_df[filtered_df['Age Group'] != i]
```

[45]:
```
filtered_df['Age Group'].unique()
```

[45]:
```
array(['0-24', '25-34', '35-44', '45-54', '55-64', '65-74', '75-84',
       '85+'], dtype=object)
```

[46]:
```
filtered_df = filtered_df[filtered_df["State"] != "United States"]
```

[47]:
```
filtered_df["State"].unique()
```

[47]:
```
array(['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',
       'Colorado', 'Connecticut', 'Delaware', 'District of Columbia',
       'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana',
       'Iowa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland',
       'Massachusetts', 'Michigan', 'Minnesota', 'Mississippi',
       'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire',
       'New Jersey', 'New Mexico', 'New York', 'New York City',
       'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
       'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota',
       'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington',
       'West Virginia', 'Wisconsin', 'Wyoming', 'Puerto Rico'],
      dtype=object)
```

[48]:
```
filtered_df.drop(columns=["Data As Of","Start Date","End␣
 ↪Date","Group","Year","Month","ICD10_codes","Flag"],inplace=True)
filtered_df
```

[48]:

	State	Condition Group	Condition	Age Group	\
230	Alabama	Respiratory diseases	Influenza and pneumonia	0-24	
231	Alabama	Respiratory diseases	Influenza and pneumonia	25-34	
232	Alabama	Respiratory diseases	Influenza and pneumonia	35-44	
233	Alabama	Respiratory diseases	Influenza and pneumonia	45-54	
234	Alabama	Respiratory diseases	Influenza and pneumonia	55-64	
…	…	…	…	…	
12413	Puerto Rico	COVID-19	COVID-19	45-54	
12414	Puerto Rico	COVID-19	COVID-19	55-64	
12415	Puerto Rico	COVID-19	COVID-19	65-74	
12416	Puerto Rico	COVID-19	COVID-19	75-84	
12417	Puerto Rico	COVID-19	COVID-19	85+	

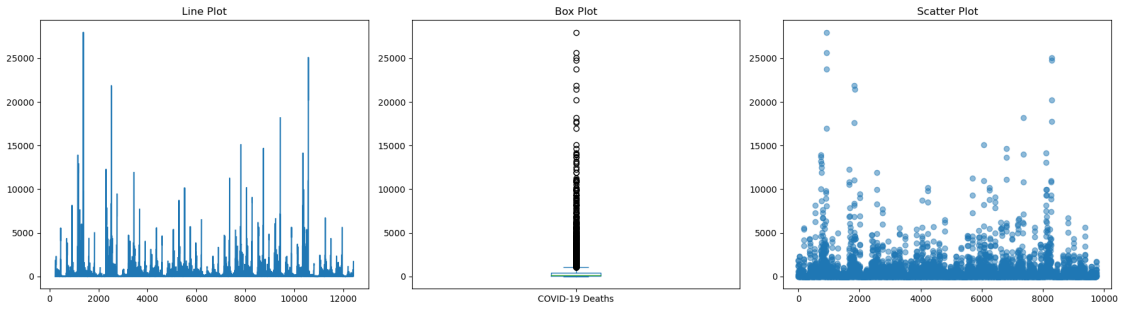
	COVID-19 Deaths	Number of Mentions
230	20.0	20.0
231	103.0	108.0
232	230.0	237.0
233	547.0	564.0

```
234              1188.0              1224.0
…                   …                   …
12413             439.0               439.0
12414             780.0               780.0
12415            1238.0              1238.0
12416            1640.0              1640.0
12417            1713.0              1713.0

[9752 rows x 6 columns]
```

[49]:
```
filtered_df.isnull().sum()
```

[49]:
```
State                    0
Condition Group          0
Condition                0
Age Group                0
COVID-19 Deaths       1646
Number of Mentions    1572
dtype: int64
```

[50]:
```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))  # 1 row, 3 columns

filtered_df["COVID-19 Deaths"].plot.line(ax=axes[0])
axes[0].set_title("Line Plot")

filtered_df["COVID-19 Deaths"].plot.box(ax=axes[1])
axes[1].set_title("Box Plot")

x_values = range(len(filtered_df))
y_values = filtered_df["COVID-19 Deaths"]
axes[2].scatter(x_values, y_values, marker='o', alpha=0.5)
axes[2].set_title("Scatter Plot")

plt.tight_layout()

plt.show()
```



[51]:
```
filtered_df.dropna(subset=["COVID-19 Deaths"],inplace=True)
```

[52]:
```
filtered_df["Number of Mentions"].fillna(method="ffill",inplace=True)
filtered_df
```

[52]:

	State	Condition Group	Condition	Age Group	\
230	Alabama	Respiratory diseases	Influenza and pneumonia	0-24	
231	Alabama	Respiratory diseases	Influenza and pneumonia	25-34	
232	Alabama	Respiratory diseases	Influenza and pneumonia	35-44	
233	Alabama	Respiratory diseases	Influenza and pneumonia	45-54	
234	Alabama	Respiratory diseases	Influenza and pneumonia	55-64	
…	…	…	…	…	
12413	Puerto Rico	COVID-19	COVID-19	45-54	
12414	Puerto Rico	COVID-19	COVID-19	55-64	
12415	Puerto Rico	COVID-19	COVID-19	65-74	
12416	Puerto Rico	COVID-19	COVID-19	75-84	
12417	Puerto Rico	COVID-19	COVID-19	85+	

	COVID-19 Deaths	Number of Mentions
230	20.0	20.0
231	103.0	108.0
232	230.0	237.0
233	547.0	564.0
234	1188.0	1224.0
…	…	…
12413	439.0	439.0
12414	780.0	780.0
12415	1238.0	1238.0
12416	1640.0	1640.0
12417	1713.0	1713.0

[8106 rows x 6 columns]

[53]:
```
age_group = {'0-24': 0,'25-34': 1,'35-44': 2,'45-54': 3,'55-64': 4,'65-74':␣
 ↪5,'75-84': 6,'85+': 7}
filtered_df["Age Group"] = filtered_df["Age Group"].map(age_group)
```

[54]:
```
condition_dict = filtered_df["Condition"].value_counts().to_dict()
two_most_frequent = sorted(condition_dict, key=condition_dict.get,␣
 ↪reverse=True)[:2]

condition_dict = {k: k if k in two_most_frequent else "Other" for k in␣
 ↪condition_dict}
filtered_df["Broad Condition Group"] = filtered_df["Condition"].
 ↪map(condition_dict)
```

[55]:
```
broad_cond_grp = {'COVID-19':0, 'All other conditions and causes (residual)':
↪1,"Other":2}
filtered_df["Broad Condition Group"] = filtered_df["Broad Condition Group"].
↪map(broad_cond_grp)
```

[56]:
```
filtered_df = filtered_df.reset_index(drop=True)
```

[57]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
filtered_df[["COVID-19 Deaths"]] = scaler.fit_transform(filtered_df[["COVID-19␣
↪Deaths"]])
final_df = filtered_df[["Age Group","COVID-19 Deaths","Broad Condition Group"]]
final_df
```

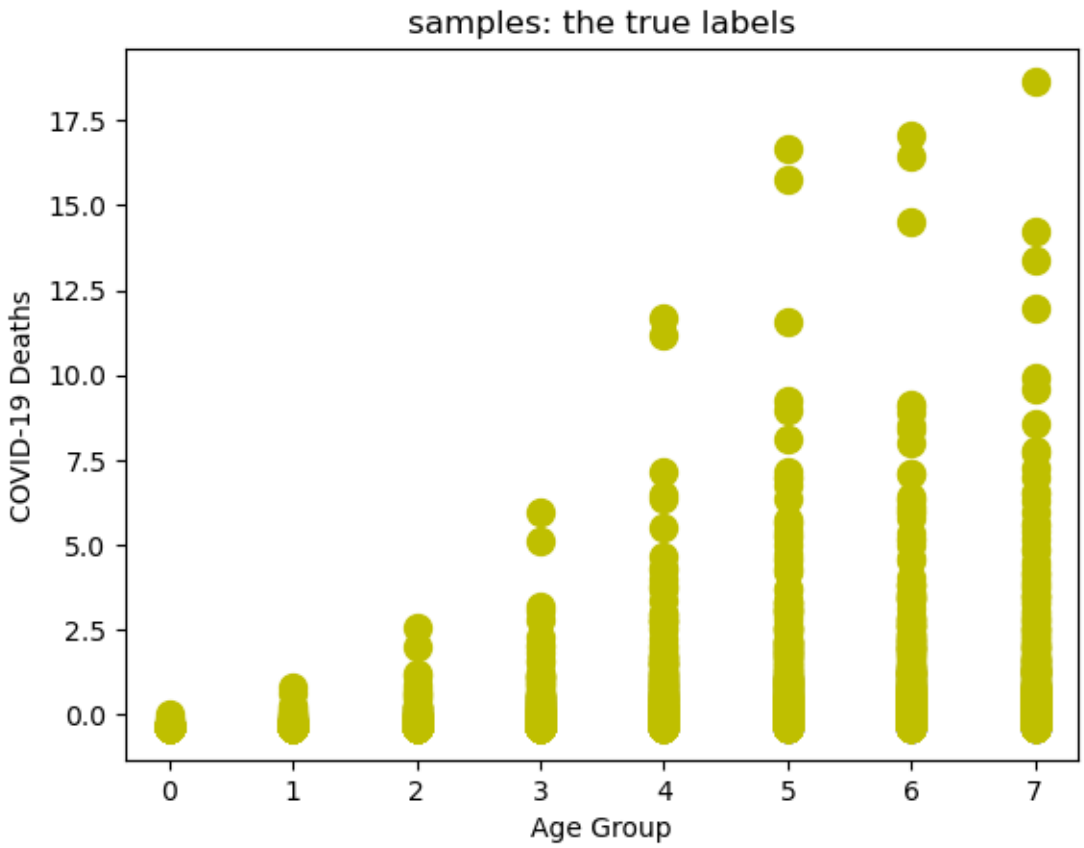
[57]:

	Age Group	COVID-19 Deaths	Broad Condition Group
0	0	-0.354732	2
1	1	-0.298298	2
2	2	-0.211946	2
3	3	0.003593	2
4	4	0.439432	2
…	…	…	…
8101	3	-0.069840	0
8102	4	0.162018	0
8103	5	0.473428	0
8104	6	0.746762	0
8105	7	0.796398	0

[8106 rows x 3 columns]

1 Defining functions for visualizing decision boundaries

[76]:
```
final_df.plot.scatter("Age Group", "COVID-19 Deaths", c="y", cmap="prism",␣
↪marker="o", s=100, colorbar=False, title="samples: the true labels");
```



2 Implementing classification algorithms and building models

2.1 Train-Test Split

[58]:
```
from sklearn.model_selection import train_test_split
X = final_df[["Age Group","COVID-19 Deaths"]]
y = final_df[["Broad Condition Group"]]

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪2,random_state=42)
```

2.2 K Nearest Neighbor

[59]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_score, recall_score, f1_score

knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train,y_train)
knn_preds = knn.predict(X_test)
```

[60]:
```
knn_precision = precision_score(y_test, knn_preds, average='weighted')
knn_recall = recall_score(y_test, knn_preds, average='weighted')
knn_f1 = f1_score(y_test, knn_preds, average='weighted')

scores_dict = {"KNN":[knn_precision,knn_recall,knn_f1]}
```

3 Logistic Regression

[61]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(class_weight="balanced")
logreg.fit(X_train,y_train)
logreg_preds = logreg.predict(X_test)
```

[62]:
```
logreg_precision = precision_score(y_test, logreg_preds, average='weighted')
logreg_recall = recall_score(y_test, logreg_preds, average='weighted')
logreg_f1 = f1_score(y_test, logreg_preds, average='weighted')

scores_dict["Logistic Regression"] = [logreg_precision,logreg_recall,logreg_f1]
```

4 SVM : Linear

[63]:
```
from sklearn.svm import SVC

svm_linear = SVC(kernel="linear", probability=True)
svm_linear.fit(X_train,y_train)
svm_linear_preds = svm_linear.predict(X_test)
```

[64]:
```
svm_linear_precision = precision_score(y_test, svm_linear_preds,␣
 ↪average='weighted')
svm_linear_recall = recall_score(y_test, svm_linear_preds, average='weighted')
svm_linear_f1 = f1_score(y_test, svm_linear_preds, average='weighted')

scores_dict["SVM Linear"] =␣
 ↪[svm_linear_precision,svm_linear_recall,svm_linear_f1]
```

5 SVM : Poly

[65]:
```
for i in [3,4,5]:
    svm_poly = SVC(kernel="poly", degree=i)
    svm_poly.fit(X_train,y_train)
    svm_poly_preds = svm_poly.predict(X_test)

    svm_poly_precision = precision_score(y_test, svm_poly_preds,␣
 ↪average='weighted')
```

```
    svm_poly_recall = recall_score(y_test, svm_poly_preds, average='weighted')
    svm_poly_f1 = f1_score(y_test, svm_poly_preds, average='weighted')

    scores_dict[f"SVM Poly {i}"] =␣
↪[svm_poly_precision,svm_poly_recall,svm_poly_f1]
```

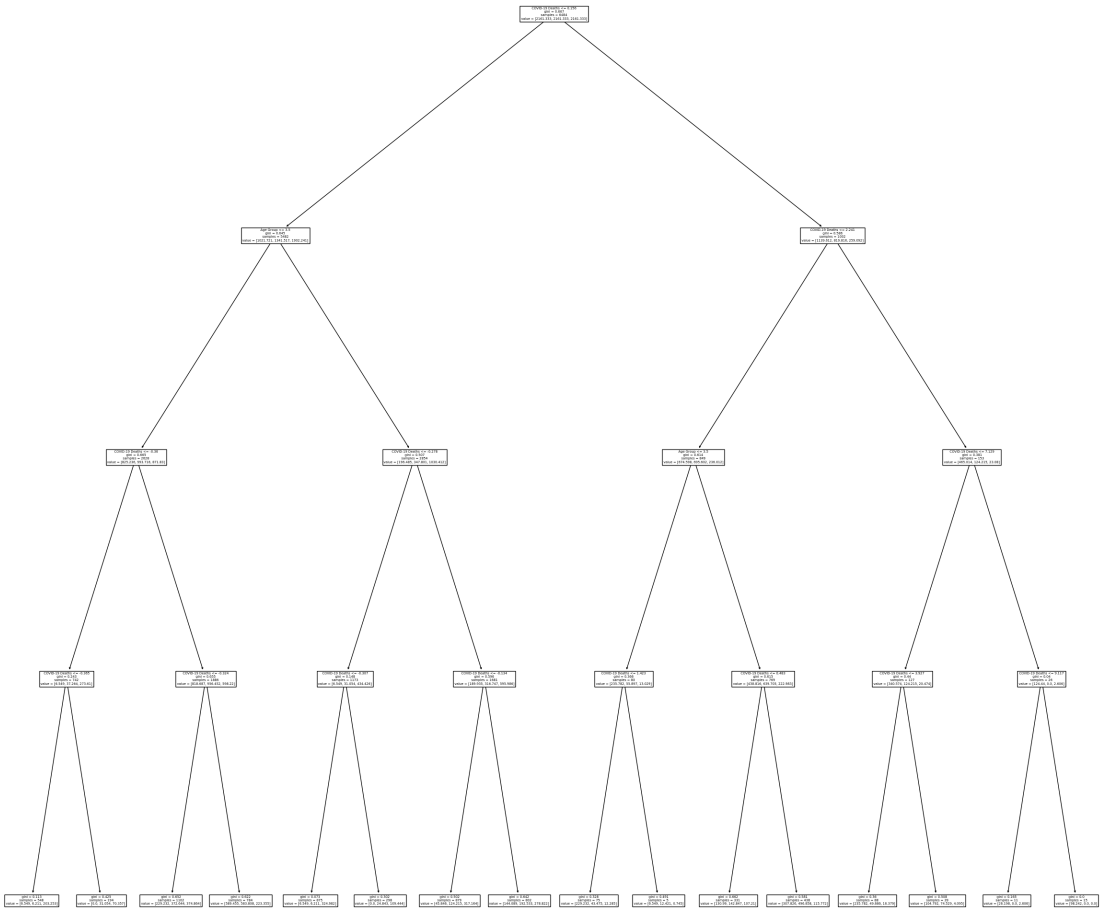
[65]:
```
{'knn': [0.8767339014987222, 0.8945745992601726, 0.8848504775905255],
 'logistic regression': [0.8892218935784733,
  0.7108508014796547,
  0.7762237836042594],
 'SVM Linear': [0.8387275588616173, 0.905055487053021, 0.8616981201653263],
 'SVM Poly 3': [0.8387275588616173, 0.905055487053021, 0.8616981201653263],
 'SVM Poly 4': [0.8387275588616173, 0.905055487053021, 0.8616981201653263],
 'SVM Poly 5': [0.8760157013394425, 0.9075215782983971, 0.8652560619711612]}
```

6 Decision Tree

[72]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

dec_tree = DecisionTreeClassifier(max_depth=4, min_samples_leaf=5,␣
 ↪random_state=42, class_weight="balanced")
dec_tree.fit(X_train,y_train)
dec_tree_preds = dec_tree.predict(X_test)
```

[67]:
```
dec_tree_precision = precision_score(y_test, dec_tree_preds, average='weighted')
dec_tree_recall = recall_score(y_test, dec_tree_preds, average='weighted')
dec_tree_f1 = f1_score(y_test, dec_tree_preds, average='weighted')

scores_dict["Decision Tree"] = [dec_tree_precision,dec_tree_recall,dec_tree_f1]
```

[74]:
```
plt.figure(figsize=(30,30))
tree.plot_tree(dec_tree, feature_names=dec_tree.feature_names_in_)
plt.show()
```



[68]:
```
scores_df = pd.DataFrame(scores_dict,index=['Precision', 'Recall', 'F1'])
scores_df
```

[68]:

	knn	logistic regression	SVM Linear	SVM Poly 3	SVM Poly 4	SVM Poly 5	Decision Tree
Precision	0.876734	0.889222	0.838728	0.838728	0.838728	0.876016	0.890258
Recall	0.894575	0.710851	0.905055	0.905055	0.905055	0.907522	0.718866
F1	0.884850	0.776224	0.861698	0.861698	0.861698	0.865256	0.783061

[ ]: