Advanced Machine Learning

CIS550

Spring ‘24

Lab Homework 5

Submitted by :

Name : Manas Vishal (01971464)

Email : mvishal@umassd.edu

Title : Deploying a model

Deploying a model

This lab is continuation of previous lab where we trained a model. In the previous lab, we used the XGBoost library to train a model on the biomedical dataset.

About the data

This biomedical dataset was built by Dr. Henrique da Mota during a medical residence period in the Group of Applied Research in Orthopaedics (GARO) of the Centre Médico-Chirurgical de Réadaptation des Massues, Lyon, France. The data has been organized in two different, but related, classification tasks.

The first task consists in classifying patients as belonging to one of three categories:

- *Normal* (100 patients)
- *Disk Hernia* (60 patients)
- *Spondylolisthesis* (150 patients)

For the second task, the categories *Disk Hernia* and *Spondylolisthesis* were merged into a single category that is labeled as *abnormal*. Thus, the second task consists in classifying patients as belonging to one of two categories: *Normal* (100 patients) or *Abnormal* (210 patients).

Each patient is represented in the dataset by six biomechanical attributes that are derived from the shape and orientation of the pelvis and lumbar spine (in this order):

- Pelvic incidence
- Pelvic tilt
- Lumbar lordosis angle
- Sacral slope
- Pelvic radius
- Grade of spondylolisthesis

The following convention is used for the class labels:

- DH (Disk Hernia)
- Spondylolisthesis (SL)
- Normal (NO)
- Abnormal (AB)

For more information about this dataset, see the [Vertebral Column dataset webpage](http://archive.ics.uci.edu/ml/datasets/Vertebral+Column).

Loading the data

We load the data by following the lab tutorial and run the commands that also loads the necessary python modules.

```
[16]
1  import pandas as pd
2  import requests
3  import zipfile
4  import io
5  from scipy.io import arff
6  from sklearn.model_selection import train_test_split
7  from xgboost import XGBClassifier
8  import warnings
9  import os
10 warnings.filterwarnings("ignore")
11 from sklearn.metrics import accuracy_score
12 import xgboost as xgb
✓ 0.0s                                                    Python
```

Fig. 1 Loading all the python modules to initialize the notebook

We then download the data and store in a dataframe variable.

```
[17]
1  f_zip = 'http://archive.ics.uci.edu/ml/machine-learning-databases/00212/vertebral_column_data.zip'
2  r = requests.get(f_zip, stream=True)
3  Vertebral_zip = zipfile.ZipFile(io.BytesIO(r.content))
4  Vertebral_zip.extractall()
5  data = arff.loadarff('column_2C_weka.arff')
6  df = pd.DataFrame(data[0])
7  class_mapper = {b'Abnormal':1,b'Normal':0}
8  df['class']=df['class'].replace(class_mapper)
9  cols = df.columns.tolist()
10 cols = cols[-1:] + cols[:-1]
11 df = df[cols]
12 train, test_and_validate = train_test_split(df, test_size=0.2, random_state=42, stratify=df['class'])
13 test, validate = train_test_split(test_and_validate, test_size=0.5, random_state=42, stratify=test_and_validate['class'])
14 model = XGBClassifier(objective='binary:logistic', eval_metric='auc', num_round=42)
15 print(model.fit(train.drop(['class'], axis = 1).values, train['class'].values))
16 print("Training Completed")
✓ 0.4s                                                    Python
```

```
[21:08:07] WARNING: /croot/xgboost-split_1675457761144/work/src/learner.cc:767:
Parameters: { "num_round" } are not used.

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='auc', feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              num_round=42, predictor=None, ...)
Training Completed
```

Fig. 2 Training the model using XGBoost

We now have to run some predictions on the deployed model. To do this, we first review the test data.

```
1  test.shape
```
[14] ✓ 0.0s Python

```
(31, 7)
```

You have 31 instances, with seven attributes. The first five instances are:

```
1  test.head(5)
```
[15] ✓ 0.0s Python

	class	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
136	1	88.024499	39.844669	81.774473	48.179830	116.601538	56.766083
230	0	65.611802	23.137919	62.582179	42.473883	124.128001	-4.083298
134	1	52.204693	17.212673	78.094969	34.992020	136.972517	54.939134
130	1	50.066786	9.120340	32.168463	40.946446	99.712453	26.766697
47	1	41.352504	16.577364	30.706191	24.775141	113.266675	-4.497958

Fig. 3 Examining the data and its features

Since, we don't have to include the target value i.e., class, we omit it by the *iloc* function. We use the usual slicing properties of python arrays.

```
1  row = test.iloc[0:1,1:]
2  row.head()
```
[5] ✓ 0.0s Python

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
136	88.024499	39.844669	81.774473	48.17983	116.601538	56.766083

Now, you can use the data to perform a prediction.

```
1  model.predict_proba(row)
```
[6] ✓ 0.0s Python

```
array([[0.00103211, 0.9989679 ]], dtype=float32)
```

Fig. 4 Removing the target value class

After the omission of the class column, we can use the remaining dataset to do a prediction. It is to be noted that we did not get a binary value instead we got a probability score.

The prediction seems to be quite accurate as the probability of model failing to classify is 0.001.

Challenge Task 1

For getting the second row, we simply changed the index of iloc from 0:1 to 1:2, and then performed the prediction.

```
1  row = test.iloc[1:2,1:]
2  row.head()
```

0.0s Python

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
230	65.611802	23.137919	62.582179	42.473883	124.128001	-4.083298

Now, you can use the data to perform a prediction.

```
1  model.predict_proba(row)
```

0.0s Python

```
array([[0.16938919, 0.8306108 ]], dtype=float32)
```

The result you get isn't a *0* or a *1*. Instead, you get a *probability score*. You can apply some conditional logic to the probability score to determine i the answer should be presented as a 0 or a 1. You will work with this process when you do batch predictions.

For now, compare the result with the test data.

```
1  test.head(5)
```

0.0s Python

	class	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
136	1	88.024499	39.844669	81.774473	48.179830	116.601538	56.766083
230	0	65.611802	23.137919	62.582179	42.473883	124.128001	-4.083298
134	1	52.204693	17.212673	78.094969	34.992020	136.972517	54.939134
130	1	50.066786	9.120340	32.168463	40.946446	99.712453	26.766697
47	1	41.352504	16.577364	30.706191	24.775141	113.266675	-4.497958

Fig. 5 Challenge taks – row 2

This prediction seems to be correct but the probability is lower so the model needs to be quantified by some metric. I am assuming that this will be covered in the next lab.

For the other rows, the model behaves similarly.

```
1  row = test.iloc[4:5,1:]
2  row.head()
```

[23] ✓ 0.0s Python

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
47	41.352504	16.577364	30.706191	24.775141	113.266675	-4.497958

Now, you can use the data to perform a prediction.

```
1  model.predict_proba(row)
```

[24] ✓ 0.0s Python

```
array([[0.02880991, 0.9711901 ]], dtype=float32)
```

Fig. 6 Challenge taks – row 5

Batch prediction

It is tedious to do the prediction on every row manually so we do a batch prediction by passing the entire dataset but without the class column.

```
1  batch_X = test.iloc[:,1:];
2  batch_X.head()
```

[8] ✓ 0.0s Python

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
136	88.024499	39.844669	81.774473	48.179830	116.601538	56.766083
230	65.611802	23.137919	62.582179	42.473883	124.128001	-4.083298
134	52.204693	17.212673	78.094969	34.992020	136.972517	54.939134
130	50.066786	9.120340	32.168463	40.946446	99.712453	26.766697
47	41.352504	16.577364	30.706191	24.775141	113.266675	-4.497958

```
1  predicted_probabilities = model.predict_proba(batch_X)
```

[9] ✓ 0.0s Python

Fig. 7 Batch prediction

We now convert the predicted probabilities into a pandas dataframe for better representation.

```
target_predicted = pd.DataFrame(predicted_probabilities[:, 1], columns=['class'])
target_predicted.head(5)
```

[10] ✓ 0.0s Python

	class
0	0.998968
1	0.830611
2	0.996538
3	0.997662
4	0.971190

Fig. 8 Dataframe of predicted probabilities

Since the class is set to the probability, we need to convert this to a binary predictor iteratively by assigning a threshold to the probability. In the figure below, we assigned 0.65 as the threshold.

```
def binary_convert(x):
    threshold = 0.65
    if x > threshold:
        return 1
    else:
        return 0

target_predicted['binary'] = target_predicted['class'].apply(binary_convert)

print(target_predicted.head(10))
test.head(10)
```

[11] ✓ 0.0s Python

```
      class  binary
0  0.998968       1
1  0.830611       1
2  0.996538       1
3  0.997662       1
4  0.971190       1
5  0.998990       1
6  0.996878       1
7  0.996293       1
8  0.997839       1
9  0.775579       1
```

	class	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
136	1	88.024499	39.844669	81.774473	48.179830	116.601538	56.766083
230	0	65.611802	23.137919	62.582179	42.473883	124.128001	-4.083298
134	1	52.204693	17.212673	78.094969	34.992020	136.972517	54.939134
130	1	50.066786	9.120340	32.168463	40.946446	99.712453	26.766697
47	1	41.352504	16.577364	30.706191	24.775141	113.266675	-4.497958
135	1	77.121344	30.349874	77.481083	46.771470	110.611148	82.093607
100	1	84.585607	30.361685	65.479486	54.223922	108.010218	25.118478
89	1	71.186811	23.896201	43.696665	47.290610	119.864938	27.283985
297	0	45.575482	18.759135	33.774143	26.816347	116.797007	3.131910
4	1	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501

Fig. 9 Binary conversion of the probabilities

Challenge Task 2

The prediction seems to be quite accurate as the probability of model is high except in the last row. I examined the output probability values and changed my threshold to 0.85 (meaning that 85% is my level of confidence). I was able to retrieve the similar trend in the target variable. However, there seems to be an edge case where the model breaks. This could be because of the model not being good which we might work in the next lab using metrics to classify a model.

```python
def binary_convert(x):
    threshold = 0.85
    if x > threshold:
        return 1
    else:
        return 0

target_predicted['binary'] = target_predicted['class'].apply(binary_convert)

print(target_predicted.head(10))
test.head(10)
```

[30] ✓ 0.0s Python

```
      class  binary
0  0.998968       1
1  0.830611       0
2  0.996538       1
3  0.997662       1
4  0.971190       1
5  0.998990       1
6  0.996878       1
7  0.996293       1
8  0.997839       1
9  0.775579       0
```

	class	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
136	1	88.024499	39.844669	81.774473	48.179830	116.601538	56.766083
230	0	65.611802	23.137919	62.582179	42.473883	124.128001	-4.083298
134	1	52.204693	17.212673	78.094969	34.992020	136.972517	54.939134
130	1	50.066786	9.120340	32.168463	40.946446	99.712453	26.766697
47	1	41.352504	16.577364	30.706191	24.775141	113.266675	-4.497958
135	1	77.121344	30.349874	77.481083	46.771470	110.611148	82.093607
100	1	84.585607	30.361685	65.479486	54.223922	108.010218	25.118478
89	1	71.186811	23.896201	43.696665	47.290610	119.864938	27.283985
297	0	45.575482	18.759135	33.774143	26.816347	116.797007	3.131910
4	1	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501

Fig. 10 Challenge task 2 -threshold 0.85

Conclusion

When deciding to apply a model in an advanced machine learning lab to a biomedical dataset, I wonder about the practical application and real-life relevance of machine learning techniques. Using the XGBoost library to train a model on Dr. Henrique da Mota's dataset was a valuable learning experience that demonstrated the power of advanced algorithms in health classification tasks.

Data classification tasks that involve classifying patients into groups. The Normal, Herniated Disc and Spondylolisthesis groups gave me a deeper understanding of the complexities of medical diagnosis. Combining Herniated Disc and Spondylolisthesis into an abnormal category for another task highlighted the need for custom models that can accommodate different classification scenarios. Six biomechanical properties derived from the pelvis and lumbar region were proven to be key features to represent patients and effectively train the model.

I participated in data exploration, model training, prediction and evaluation and gained practical experience during the laboratory exercises. in the process of machine learning. Analyzing the probability scores and adjusting the prediction thresholds provided me with valuable information about model performance and the delicate balance between sensitivity and specificity. Identifying edge cases in modeling problems emphasized the importance of continuous improvement of reliable model estimation and machine learning algorithms.

In addition, challenges in predicting certain cases motivated me to delve deeper into model refinement and improvement. explore advanced

features. assessment techniques. By focusing on improving model evaluation metrics and optimizing performance strategies, I aim to improve the reliability and generalizability of my machine learning model. This iterative process of model improvement and evaluation is necessary to ensure the effectiveness of machine learning applications in real healthcare environments.

Finally, implementing a model in an advanced machine learning lab not only gave me practical skills. in model implementation and evaluation, but also deepened my understanding of the complexities of applying machine learning to biomedical datasets. Combining theory with hands-on practices, this laboratory exercise laid a strong foundation for future advanced machine learning techniques and their transformative potential in healthcare and research.