Introduction

Up to this point, we've covered the foundational concepts of machine learning. You may have already experimented with various hyperparameters to optimize model performance. In this notebook, you'll be introduced to several advanced techniques designed to further enhance your models.

This notebook consists the following parts:

Learning Objectives

By the end of this two weeks you will be able to:

- Understand the fundamental concepts of ensemble learning.
- Use evaluation techniques to assess models performance.
- Enhance model performance by feature engineering.
- Build pipelines for model development and preprocessing

Instructions

- Ensure you fully understand the requirements and objectives of the assignment.
- Review the notebooks refered in the tasks
- If you need additional context or clarification, please check the provided videos or background literature.
- Work through each part of the assignment methodically, ensuring all tasks are completed.
- Update your repository with your new created work

Additional Notes:

- Do not add datafiles to your repository. Repositories with datafiles will not be accepted
- Class solutions should be delivered in python files. Not in notebooks
- When AI tools are used, you must provide proper references and explanations for how they were utilized. Failure to do so will be considered as academic fraud
- The bonus assignment are not mandatory

- Use PEP8

Good luck!

F.Feenstra

Part A. Data retrieval

The dataset you can use for this notebook is the lung dataset from Maastricht University. It comprises 89 non-small cell lung cancer (NSCLC) patients records who underwent surgical treatment. The study where the data is from explored the relationship between radiomic imaging features and gene expression profiles. The samples were collected through biopsies at MAASTRO Clinic in The Netherlands, and the dataset is publicly available.

The authors of the related paper discovered that a prognostic radiomic signature, which captures intra-tumor heterogeneity, is closely associated with underlying gene expression patterns. Developing a machine learning model to predict histology from the Clinical and Genetic Lung data can improve diagnostic accuracy and treatment personalization. In this notebook we will develop such a prediction model.

Availabel Datasets:

- Lung metadata dataset [1]
- Gene expression dataset [2]

Important It is also allowed to use your own dataset from your own project if this data is highly dimensional and contains genetic information.

[1] NSCLC-Radiomics-Genomics

[2] Gene Expression Data - GSE58661

[3] Aerts HJWL, Rios Velazquez E, Leijenaar RTH, Parmar C, Grossmann P, Carvalho S, Bussink J, Monshouwer R, Haibe-Kains B, Rietveld D, Hoebers F, Rietbergen MM, Leemans CR, Dekker A, Quackenbush J, Gillies RJ, & Lambin P. (2015). Data From NSCLC-Radiomics-Genomics. The Cancer Imaging Archive. https://doi.org/10.7937/K9/TCIA.2015.L4FRET6Z

Data retrieval task

- Retrieve the data. (No cleaning needed yet)

```
import pandas as pd
import gzip

lung_pd = pd.read_excel('Lung3.metadata.xls')
lung_pd.rename(columns=lambda x: x.replace('characteristics.tag.',
''), inplace=True)
print(lung_pd.columns)
cols = ['title', 'histology', 'tumor.size.maximumdiameter',
```

```
'stage.primary.tumor', 'gender']
lung_pd = lung_pd[cols]
lung_pd.dropna(inplace=True)
print(lung_pd.columns)
print(lung_pd.head())

lung_pd['histology'] = lung_pd['histology'].apply(lambda x:
x.replace('\xa0', ' ').lower().strip())
uniques = set(list(lung_pd['histology']))


data_gse = pd.read_csv('GSE58661_series_matrix.txt.gz', sep='\t',
comment='!', quotechar='"', compression='gzip')
data_gse.dropna(inplace=True)
data_gse.set_index("ID_REF", inplace=True)
print(data_gse.columns)
data_gse.head()
```

```
Index(['sample.name', 'title', 'CEL.file', 'source.location',
'organism',
       'gender', 'histology', 'tumor.size.maximumdiameter',
       'stage.primary.tumor', 'stage.nodes', 'stage.mets',
'primaryVSmets',
       'grade', 'molecule tested', 'label', 'platform'],
      dtype='object')
Index(['title', 'histology', 'tumor.size.maximumdiameter',
       'stage.primary.tumor', 'gender'],
      dtype='object')
    title                        histology  tumor.size.maximumdiameter
\
0  lung_1    Squamous Cell Carcinoma, NOS                          4.0

1  lung_2  Adenocarcinoma, Papillary, NOS                          1.3

2  lung_3                  Non-Small Cell                         11.0

4  lung_5    Squamous Cell Carcinoma, NOS                          7.8

5  lung_6             Adenocarcinoma, NOS                          3.5


  stage.primary.tumor gender
0                 pT2      M
1                 pT1      M
2                 pT3      M
4                 pT3      F
5                 pT2      M
Index(['GSM1416528', 'GSM1416529', 'GSM1416530', 'GSM1416531',
'GSM1416532',
       'GSM1416533', 'GSM1416534', 'GSM1416535', 'GSM1416536',
```

```
'GSM1416537',
       'GSM1416538', 'GSM1416539', 'GSM1416540', 'GSM1416541',
'GSM1416542',
       'GSM1416543', 'GSM1416544', 'GSM1416545', 'GSM1416546',
'GSM1416547',
       'GSM1416548', 'GSM1416549', 'GSM1416550', 'GSM1416551',
'GSM1416552',
       'GSM1416553', 'GSM1416554', 'GSM1416555', 'GSM1416556',
'GSM1416557',
       'GSM1416558', 'GSM1416559', 'GSM1416560', 'GSM1416561',
'GSM1416562',
       'GSM1416563', 'GSM1416564', 'GSM1416565', 'GSM1416566',
'GSM1416567',
       'GSM1416568', 'GSM1416569', 'GSM1416570', 'GSM1416571',
'GSM1416572',
       'GSM1416573', 'GSM1416574', 'GSM1416575', 'GSM1416576',
'GSM1416577',
       'GSM1416578', 'GSM1416579', 'GSM1416580', 'GSM1416581',
'GSM1416582',
       'GSM1416583', 'GSM1416584', 'GSM1416585', 'GSM1416586',
'GSM1416587',
       'GSM1416588', 'GSM1416589', 'GSM1416590', 'GSM1416591',
'GSM1416592',
       'GSM1416593', 'GSM1416594', 'GSM1416595', 'GSM1416596',
'GSM1416597',
       'GSM1416598', 'GSM1416599', 'GSM1416600', 'GSM1416601',
'GSM1416602',
       'GSM1416603', 'GSM1416604', 'GSM1416605', 'GSM1416606',
'GSM1416607',
       'GSM1416608', 'GSM1416609', 'GSM1416610', 'GSM1416611',
'GSM1416612',
       'GSM1416613', 'GSM1416614', 'GSM1416615', 'GSM1416616'],
      dtype='object')

                GSM1416528  GSM1416529  GSM1416530  GSM1416531
GSM1416532  \
ID_REF

AFFX-BioB-3_at    7.376915    8.024915    7.522543    7.152864
7.211031
AFFX-BioB-5_at    6.984530    7.427048    7.077207    6.849513
6.753131
AFFX-BioB-M_at    7.330576    8.010530    7.334551    7.143286
7.077163
AFFX-BioC-3_at   10.922741   11.390638   10.936703   10.791909
10.701328
AFFX-BioC-5_at   11.032030   11.533338   11.018510   10.875259
10.823792

                GSM1416533  GSM1416534  GSM1416535  GSM1416536
```

```
GSM1416537  \
ID_REF

AFFX-BioB-3_at    7.704962    7.779374    7.136893    7.160221
8.141578
AFFX-BioB-5_at    7.195947    7.429800    6.745630    6.751358
7.632631
AFFX-BioB-M_at    7.753484    7.712429    7.036303    7.089186
7.992361
AFFX-BioC-3_at   11.221732   11.084229   10.588372   10.628909
11.412668
AFFX-BioC-5_at   11.276973   11.178643   10.700420   10.779332
11.470336

                ...  GSM1416607  GSM1416608  GSM1416609  GSM1416610  \
ID_REF          ...
AFFX-BioB-3_at  ...    7.611992    7.345206    7.316975    7.418340
AFFX-BioB-5_at  ...    7.175015    7.014479    6.913879    6.898449
AFFX-BioB-M_at  ...    7.612393    7.285821    7.324635    7.408880
AFFX-BioC-3_at  ...   11.097486   10.908682   10.781228   10.935723
AFFX-BioC-5_at  ...   11.265739   10.990373   10.900765   11.101546

                GSM1416611  GSM1416612  GSM1416613  GSM1416614
GSM1416615  \
ID_REF

AFFX-BioB-3_at    7.756286    7.215856    7.465109    7.245458
7.039592
AFFX-BioB-5_at    7.347570    6.740992    7.080787    6.725812
6.575376
AFFX-BioB-M_at    7.745624    7.152722    7.422373    7.263596
6.975162
AFFX-BioC-3_at   11.200635   10.707980   10.736531   10.770440
10.367009
AFFX-BioC-5_at   11.343329   10.842697   10.900509   10.838008
10.494806

                GSM1416616
ID_REF
AFFX-BioB-3_at    7.393667
AFFX-BioB-5_at    7.041106
AFFX-BioB-M_at    7.366620
AFFX-BioC-3_at   10.753221
AFFX-BioC-5_at   10.863600

[5 rows x 89 columns]
```

Part B. Feature engineering

Mind you that choosing an algorithm and hyperparameter tuning might not be enough. If your data is of low quality, the algorithm will have poor performance as well. This is where feature engineering comes into play. Feature engineering involves transforming raw data into meaningful features that better represent the underlying problem to the predictive models, ultimately enhancing the model's performance. By selecting, creating, and refining features, you ensure that your data highlights the most relevant patterns and relationships, allowing the algorithm to learn more effectively. Proper feature engineering can often make the difference between a moderate model and a highly accurate one, even more so than the choice of algorithm itself. Possible modifications:

- creation of new features derived from original features
- selection of features
- encoding features
- log transformation
- scaling
- dimension reduction (*e.g.* PCA)

Feature engineering Task

- Review the Study Case Feature Engineering notebook.
- Review the Study Case RNA-seq Preparation notebook.
- Assess which data preparation and feature engineering steps could be beneficial for your dataset.
- Implement a `DataProcessor` class tailored to your dataset and test it in the cell below. **Make sure it's an appropriate sklearn object, such as a `Transformer` or `Pipeline`.**
- Update your repository with a new directory named `optimization`, including the following:
  - The `DataProcessor` class as a Python file, complete with thorough documentation.
  - An evaluation document that details and justifies your choices using a well-reasoned, argumentative approach.

```
# YOUR CODE HERE TO DEMONSTRATE THE USAGE OF THE PREPROCESSOR CLASS
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import numpy as np

data_gse_trans = data_gse.transpose()

# To combat skewed data
data_gse_log = np.log1p(data_gse_trans)

#Standardizeing since genes can have very different scales of
expression.
scaler = StandardScaler()
scaled_gse_trans_log = scaler.fit_transform(data_gse_log)
```

```python
# Performing PCA I know there is a better way to do like doing a pca
on all data and then look at what point it becomes more than 0.5
total.
# But this is to practice recursion.
def find_components(data, start=2):
    pca = PCA(n_components=start)
    pca_res = pca.fit_transform(data)
    expl_var_list = pca.explained_variance_ratio_

    if sum(expl_var_list) >= 0.5:
        print(f"N component to reach 50% explained of the variance:
{start}")
        print(f"Explained variance list: {expl_var_list}")
        return start, pca, expl_var_list, pca_res
    else:
        start += 1
        return find_components(data, start)


start, pca, expl_var_list, pca_res =
find_components(scaled_gse_trans_log)
columns1 = [f"PCA_{i+1}" for i in range(start)]


pca_res_df = pd.DataFrame(data=pca_res, columns=columns1)

merged_df = pd.merge(lung_pd, pca_res_df, left_index=True,
right_index=True)
print(merged_df.head())
```

```
N component to reach 50% explained of the variance: 16
Explained variance list: [0.09358163 0.08701857 0.04763506 0.0439992
0.03330682 0.02687025
 0.02465527 0.02241783 0.01850185 0.01725881 0.01621868 0.01531552
 0.01433772 0.01354639 0.01325415 0.01315874]
    title                       histology  tumor.size.maximumdiameter
\
0  lung_1    squamous cell carcinoma, nos                          4.0

1  lung_2  adenocarcinoma, papillary, nos                          1.3

2  lung_3                 non-small cell                         11.0

4  lung_5    squamous cell carcinoma, nos                          7.8

5  lung_6             adenocarcinoma, nos                          3.5


  stage.primary.tumor gender      PCA_1       PCA_2      PCA_3
PCA_4  \
0                 pT2      M -12.563698  143.350916 -74.113707 -
```

```
29.447517
1                  pT1      M -14.585510 -136.108180 -31.682665 -
93.293509
2                  pT3      M -72.359515   31.080436  -7.212311 -
23.003114
4                  pT3      F -61.898539   -2.743114 -34.230067
42.663674
5                  pT2      M  17.562691  -86.546114 -10.003988
41.723127

       PCA_5  ...      PCA_7      PCA_8      PCA_9     PCA_10
PCA_11  \
0 -22.873815  ...  32.850302 -33.709224  62.696240  -4.411993
19.968412
1 -35.638663  ...  13.783358   6.443782  -3.767860  52.725018
55.846064
2  -3.620890  ... -45.965215 -77.529470  83.729802  -1.545160
119.487783
4  44.716229  ... -50.697751   1.434348 -16.768901 -27.776140  -
22.891227
5  17.580223  ... -24.641162   0.501947  26.509949 -18.814753
5.222445

      PCA_12     PCA_13     PCA_14     PCA_15      PCA_16
0 -11.276013   6.471942  -0.184468 -22.009904    5.381011
1  -2.812119  -3.451776  81.455464  12.315144 -109.935105
2   7.006954 -42.027303  -7.773518 -64.385517   54.684574
4 -16.557359 -26.567846  29.651508 -16.065088  -10.076500
5  21.309420   2.921614   8.611129 -11.747633   -9.636650

[5 rows x 21 columns]
```

```
# Encoder used for turning strings/classifications into numeric values
from sklearn.preprocessing import LabelEncoder

def encode_column(df, column_enc):
    encoder = LabelEncoder()
    df[column_enc] = encoder.fit_transform(df[column_enc])
    return df

merged_df = encode_column(merged_df, 'histology')
merged_df = encode_column(merged_df, 'stage.primary.tumor')

from DataProcessor import DataProcessor

pipeline = DataProcessor(input_file_exp =
'GSE58661_series_matrix.txt.gz', input_file_meta =
"Lung3.metadata.xls",
                         cols_to_grab = ['title',
       'characteristics.tag.gender', 'characteristics.tag.histology',
```

```
       'characteristics.tag.tumor.size.maximumdiameter'],
target='characteristics.tag.histology')

pipeline_finished, encoders = pipeline.pipeline_run()
print(encoders)
print(pipeline_finished.head())
print(pipeline.look_at_encode(encoders, "title"))
```

```
N component to reach 50% explained of the variance: 14
Explained variance list: [0.13625197 0.0657839  0.04725548 0.04508164
0.03637912 0.02774053
 0.02481753 0.02132643 0.02092515 0.01706445 0.01596801 0.01559264
 0.01437887 0.01392811]
['title', 'gender', 'histology']
{'title': LabelEncoder(), 'gender': LabelEncoder()}
   title  gender                         histology
tumor.size.maximumdiameter  \
0      0       1    Squamous Cell Carcinoma, NOS
4.0
1     11       1  Adenocarcinoma, Papillary, NOS
1.3
2     22       1                  Non-Small Cell
11.0
4     43       0    Squamous Cell Carcinoma, NOS
7.8
5     54       1             Adenocarcinoma, NOS
3.5

       PCA_1     PCA_2     PCA_3      PCA_4     PCA_5     PCA_6
PCA_7  \
0  15.075208 -5.472546  0.145589   2.995557  3.196469 -0.816480 -
8.442277
1  -6.797299  9.107596 -4.149766  10.091782 -3.867643  2.197285 -
0.667493
2   7.656499  5.641671  2.394907   0.509648 -0.128076  7.513621 -
9.243774
4   6.930568 -1.981705 -3.092332  -2.864091 -4.972384  0.097903
3.851141
5  -6.866977  0.155878 -4.951343  -4.213497  0.500106 -1.366388
0.556406

      PCA_8     PCA_9    PCA_10    PCA_11    PCA_12     PCA_13
PCA_14
0  2.144322  0.371191 -1.495906 -4.127096 -0.762431  -3.680627 -
1.920025
1  2.812155 -2.059079 -2.073999  5.962730  0.650969   3.316948 -
5.870156
2  5.257675 -4.131223  5.760677 -8.454589  1.021437  12.263974
```

```
9.003247
4 -4.317892  4.827766  1.086183  1.126033  3.397036   1.218709
3.900536
5 -2.570907 -0.207244  0.158308 -1.783719 -2.991035   5.627051 -
0.053776
{'lung_1': 0, 'lung_10': 1, 'lung_11': 2, 'lung_12': 3, 'lung_13': 4,
'lung_14': 5, 'lung_15': 6, 'lung_16': 7, 'lung_17': 8, 'lung_18': 9,
'lung_19': 10, 'lung_2': 11, 'lung_20': 12, 'lung_21': 13, 'lung_22':
14, 'lung_23': 15, 'lung_24': 16, 'lung_25': 17, 'lung_26': 18,
'lung_27': 19, 'lung_28': 20, 'lung_29': 21, 'lung_3': 22, 'lung_30':
23, 'lung_31': 24, 'lung_32': 25, 'lung_33': 26, 'lung_34': 27,
'lung_35': 28, 'lung_36': 29, 'lung_37': 30, 'lung_38': 31, 'lung_39':
32, 'lung_40': 33, 'lung_41': 34, 'lung_42': 35, 'lung_43': 36,
'lung_44': 37, 'lung_45': 38, 'lung_46': 39, 'lung_47': 40, 'lung_48':
41, 'lung_49': 42, 'lung_5': 43, 'lung_50': 44, 'lung_51': 45,
'lung_52': 46, 'lung_53': 47, 'lung_54': 48, 'lung_55': 49, 'lung_56':
50, 'lung_57': 51, 'lung_58': 52, 'lung_59': 53, 'lung_6': 54,
'lung_60': 55, 'lung_61': 56, 'lung_62': 57, 'lung_63': 58, 'lung_64':
59, 'lung_65': 60, 'lung_66': 61, 'lung_67': 62, 'lung_68': 63,
'lung_69': 64, 'lung_7': 65, 'lung_70': 66, 'lung_71': 67, 'lung_72':
68, 'lung_73': 69, 'lung_74': 70, 'lung_75': 71, 'lung_76': 72,
'lung_77': 73, 'lung_78': 74, 'lung_79': 75, 'lung_8': 76, 'lung_80':
77, 'lung_81': 78, 'lung_82': 79, 'lung_83': 80, 'lung_84': 81,
'lung_85': 82, 'lung_86': 83, 'lung_87': 84, 'lung_88': 85, 'lung_89':
86, 'lung_9': 87}
{'lung_1': 0, 'lung_10': 1, 'lung_11': 2, 'lung_12': 3, 'lung_13': 4,
'lung_14': 5, 'lung_15': 6, 'lung_16': 7, 'lung_17': 8, 'lung_18': 9,
'lung_19': 10, 'lung_2': 11, 'lung_20': 12, 'lung_21': 13, 'lung_22':
14, 'lung_23': 15, 'lung_24': 16, 'lung_25': 17, 'lung_26': 18,
'lung_27': 19, 'lung_28': 20, 'lung_29': 21, 'lung_3': 22, 'lung_30':
23, 'lung_31': 24, 'lung_32': 25, 'lung_33': 26, 'lung_34': 27,
'lung_35': 28, 'lung_36': 29, 'lung_37': 30, 'lung_38': 31, 'lung_39':
32, 'lung_40': 33, 'lung_41': 34, 'lung_42': 35, 'lung_43': 36,
'lung_44': 37, 'lung_45': 38, 'lung_46': 39, 'lung_47': 40, 'lung_48':
41, 'lung_49': 42, 'lung_5': 43, 'lung_50': 44, 'lung_51': 45,
'lung_52': 46, 'lung_53': 47, 'lung_54': 48, 'lung_55': 49, 'lung_56':
50, 'lung_57': 51, 'lung_58': 52, 'lung_59': 53, 'lung_6': 54,
'lung_60': 55, 'lung_61': 56, 'lung_62': 57, 'lung_63': 58, 'lung_64':
59, 'lung_65': 60, 'lung_66': 61, 'lung_67': 62, 'lung_68': 63,
'lung_69': 64, 'lung_7': 65, 'lung_70': 66, 'lung_71': 67, 'lung_72':
68, 'lung_73': 69, 'lung_74': 70, 'lung_75': 71, 'lung_76': 72,
'lung_77': 73, 'lung_78': 74, 'lung_79': 75, 'lung_8': 76, 'lung_80':
77, 'lung_81': 78, 'lung_82': 79, 'lung_83': 80, 'lung_84': 81,
'lung_85': 82, 'lung_86': 83, 'lung_87': 84, 'lung_88': 85, 'lung_89':
86, 'lung_9': 87}
```

Part C. Model Evaluation

Evaluating the performance of a machine learning model goes beyond just looking at accuracy, as accuracy alone can be misleading, especially in cases where the dataset is imbalanced or where different types of errors have different consequences

- **Detecting Overfitting/Underfitting**: A learning curve can help you understand whether your model is overfitting (performing well on training data but poorly on validation data) or underfitting (performing poorly on both training and validation data)
- **ROC**: The ROC curve shows how well your model distinguishes between classes. It helps in selecting the optimal threshold for classification decisions, particularly when the cost of false positives and false negatives differs significantly. ROC curves are often used to compare models.
- **Confusion matrix**: From the confusion matrix, you can derive other important metrics like precision, recall, F1-score, and specificity, which give a better understanding of how your model is performing across different classes

Evaluation Task

- Review the documentation for `sklearn.metrics` and `learning_curve`.
- Select appropriate metrics for your dataset, including accuracy and indicators of overfitting or underfitting.
- Implement functions to compute and assess these metrics. It is allowed to use libraries.
- Add the evaluation functions as a Python module to your repository.
- Update the evaluation documentation to clearly explain your choices for the evaluation metrics and the rationale behind them.

See also: Model evaluation video

Part D. Ensemble learning

Ensemble learning is a powerful machine learning technique that combines the predictions of multiple models to improve overall performance, robustness, and accuracy. Rather than relying on a single model, ensemble learning methods aggregate the results of several models—often called "weak learners"—to produce a stronger predictive model. The key idea behind ensemble learning is that by combining models, the weaknesses of individual models can be offset, leading to better generalization on unseen data. Popular ensemble ML algorithms are the `Random Forest` and `XGBoost`. Here's an improved version of the introduction:

voting algorithms

Voting algorithms in ensemble learning combine the predictions of multiple classifiers to make a final decision, typically based on the consensus or weighted agreement among models. Two primary types of voting are commonly used: hard voting, where the final prediction is

determined by the majority vote, and soft voting, which uses the weighted average of predicted probabilities to determine the outcome. Several sites explain the hard and soft voting algorithm. A clear explanation can be found on https://www.baeldung.com/cs/hard-vs-soft-voting-classifiers

Ensemble Task

- Review the Study Case notebook on ensemble learning.
- Try three different algorithms for classification of your data label.
- Implement a hard and soft voting algorithm for model aggregation.
- Compare the performance of the voting algorithm with that of a boosting or bagging algorithm.
- Update your repository with the voting algorithm class

See also: ensemle learning video

```
# YOUR CODE HERE WITH THE VOTING ALGORITHMS
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report,
precision_score, confusion_matrix, roc_curve, roc_auc_score,
recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from numpy import mean

print(pipeline_finished.columns)
print(pipeline_finished.head())

svccl = SVC(kernel='sigmoid')
knncl = KNeighborsClassifier(n_neighbors=5)
rfcl = RandomForestClassifier(random_state=29)

models = {"svccl": svccl, "knncl":knncl, "rfcl":rfcl}

y = pipeline_finished['histology']
X = pipeline_finished.drop(columns=['histology', 'title', 'gender'])

print(len(X))

rare_class = y.value_counts()[y.value_counts() < 5].index
X = X[~y.isin(rare_class)]
y = y[~y.isin(rare_class)]
print(len(X))


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=20)

svccl.fit(X_train, y_train)
```

```
knncl.fit(X_train, y_train)
rfcl.fit(X_train, y_train)

svccl_pred = svccl.predict(X_test)
knncl_pred = knncl.predict(X_test)
rfcl_pred = rfcl.predict(X_test)


def get_scores(model_name, y_test, model, X, y):

    model.fit(X_train, y_train)
    model_pred = model.predict(X_test)

    print(f"\n\n\nModel name: {model_name}")
    print(f"Precision: {precision_score(y_test, model_pred,
average='weighted', zero_division=0)}")
    print(f"Accuracy: {accuracy_score(y_test, model_pred)}")
    print(f"Recall: {recall_score(y_test, model_pred,
average='weighted', zero_division=0)}")
    print(f"F1 Score: {f1_score(y_test, model_pred,
average='weighted', zero_division=0)}")

    cross_score = cross_val_score(model, X, y, cv=5)
    print(f"Cross_scores:\nCross val mean: {mean(cross_score)}")
    print(f"Cross val list: {cross_score}")

    print(f"ConfusionMatrix:\n Matrix: {confusion_matrix(y_test,
model_pred)}")
    print(f"Classification report:\nReport
{classification_report(y_test, model_pred, zero_division=0)}")
    return model, model_pred

for model in models:
    get_scores(f"{model}", y_test, models[model], X, y)
    print(model)



Index(['title', 'gender', 'histology', 'tumor.size.maximumdiameter',
'PCA_1',
       'PCA_2', 'PCA_3', 'PCA_4', 'PCA_5', 'PCA_6', 'PCA_7', 'PCA_8',
'PCA_9',
       'PCA_10', 'PCA_11', 'PCA_12', 'PCA_13', 'PCA_14'],
      dtype='object')
   title  gender                        histology
tumor.size.maximumdiameter  \
0      0       1    Squamous Cell Carcinoma, NOS
4.0
1     11       1  Adenocarcinoma, Papillary, NOS
1.3
```

```
2     22       1                  Non-Small Cell
11.0
4     43       0    Squamous Cell Carcinoma, NOS
7.8
5     54       1             Adenocarcinoma, NOS
3.5

       PCA_1     PCA_2     PCA_3      PCA_4     PCA_5     PCA_6
PCA_7  \
0  15.075208 -5.472546  0.145589   2.995557  3.196469 -0.816480 -
8.442277
1  -6.797299  9.107596 -4.149766  10.091782 -3.867643  2.197285 -
0.667493
2   7.656499  5.641671  2.394907   0.509648 -0.128076  7.513621 -
9.243774
4   6.930568 -1.981705 -3.092332  -2.864091 -4.972384  0.097903
3.851141
5  -6.866977  0.155878 -4.951343  -4.213497  0.500106 -1.366388
0.556406

      PCA_8     PCA_9    PCA_10    PCA_11    PCA_12     PCA_13
PCA_14
0  2.144322  0.371191 -1.495906 -4.127096 -0.762431  -3.680627 -
1.920025
1  2.812155 -2.059079 -2.073999  5.962730  0.650969   3.316948 -
5.870156
2  5.257675 -4.131223  5.760677 -8.454589  1.021437  12.263974
9.003247
4 -4.317892  4.827766  1.086183  1.126033  3.397036   1.218709
3.900536
5 -2.570907 -0.207244  0.158308 -1.783719 -2.991035   5.627051 -
0.053776
88
65



Model name: svccl
Precision: 0.6791666666666667
Accuracy: 0.8
Recall: 0.8
F1 Score: 0.7345454545454546
Cross_scores:
Cross val mean: 0.8153846153846155
Cross val list: [0.84615385 0.76923077 0.84615385 0.84615385
0.76923077]
ConfusionMatrix:
 Matrix: [[10  0  0]
 [ 1  0  2]
 [ 1  0  6]]
```

```
Classification report:
Report                                precision    recall  f1-score
support

         Adenocarcinoma, NOS       0.83      1.00      0.91        10
              Non-Small Cell       0.00      0.00      0.00         3
Squamous Cell Carcinoma, NOS       0.75      0.86      0.80         7

                    accuracy                           0.80        20
                   macro avg       0.53      0.62      0.57        20
                weighted avg       0.68      0.80      0.73        20

svccl



Model name: knncl
Precision: 0.6424242424242423
Accuracy: 0.75
Recall: 0.75
F1 Score: 0.6910714285714286
Cross_scores:
Cross val mean: 0.7692307692307694
Cross val list: [0.76923077 0.69230769 0.84615385 0.84615385
0.69230769]
ConfusionMatrix:
 Matrix: [[9 0 1]
 [1 0 2]
 [1 0 6]]
Classification report:
Report                                precision    recall  f1-score
support

         Adenocarcinoma, NOS       0.82      0.90      0.86        10
              Non-Small Cell       0.00      0.00      0.00         3
Squamous Cell Carcinoma, NOS       0.67      0.86      0.75         7

                    accuracy                           0.75        20
                   macro avg       0.49      0.59      0.54        20
                weighted avg       0.64      0.75      0.69        20

knncl



Model name: rfcl
Precision: 0.5186813186813187
Accuracy: 0.55
Recall: 0.55
F1 Score: 0.5041176470588236
```

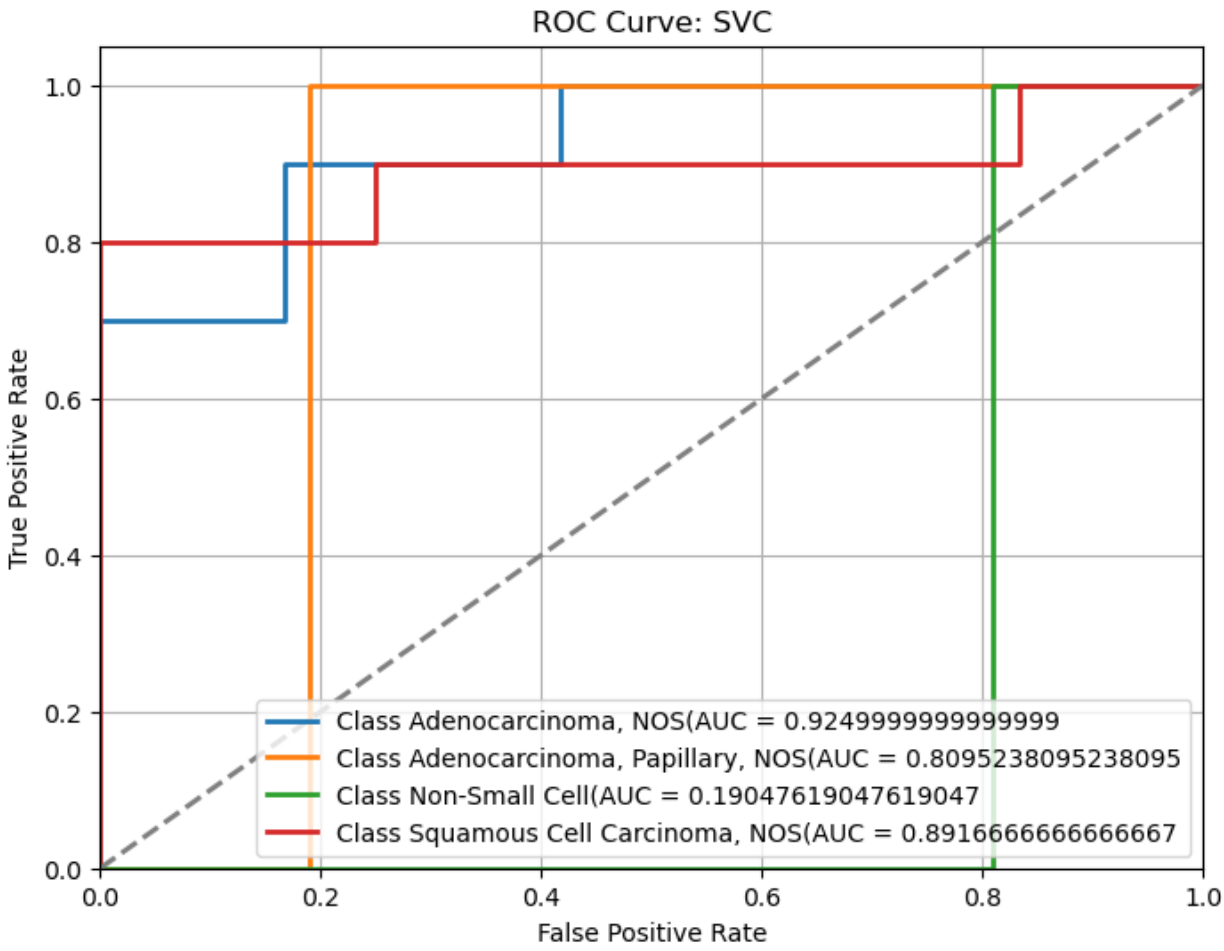
```
Cross_scores:
Cross val mean: 0.7538461538461538
Cross val list: [0.84615385 0.69230769 0.84615385 0.69230769
0.69230769]
ConfusionMatrix:
 Matrix: [[5 0 5]
 [1 0 2]
 [1 0 6]]
Classification report:
Report                                precision    recall  f1-score
support

         Adenocarcinoma, NOS       0.71      0.50      0.59        10
              Non-Small Cell       0.00      0.00      0.00         3
Squamous Cell Carcinoma, NOS       0.46      0.86      0.60         7

                    accuracy                           0.55        20
                   macro avg       0.39      0.45      0.40        20
                weighted avg       0.52      0.55      0.50        20
```

```
rfcl
```

```
from Evaluation import Evaluator
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

svccl = SVC(kernel='sigmoid', probability=True)
knncl = KNeighborsClassifier(n_neighbors=5)
rfcl = RandomForestClassifier(random_state=29)

models = [svccl, knncl, rfcl]

y = pipeline_finished['histology']
X = pipeline_finished.drop(columns=['histology', 'title', 'gender'])

eval_class = Evaluator(models, X, y)

eval_class.execute()
```

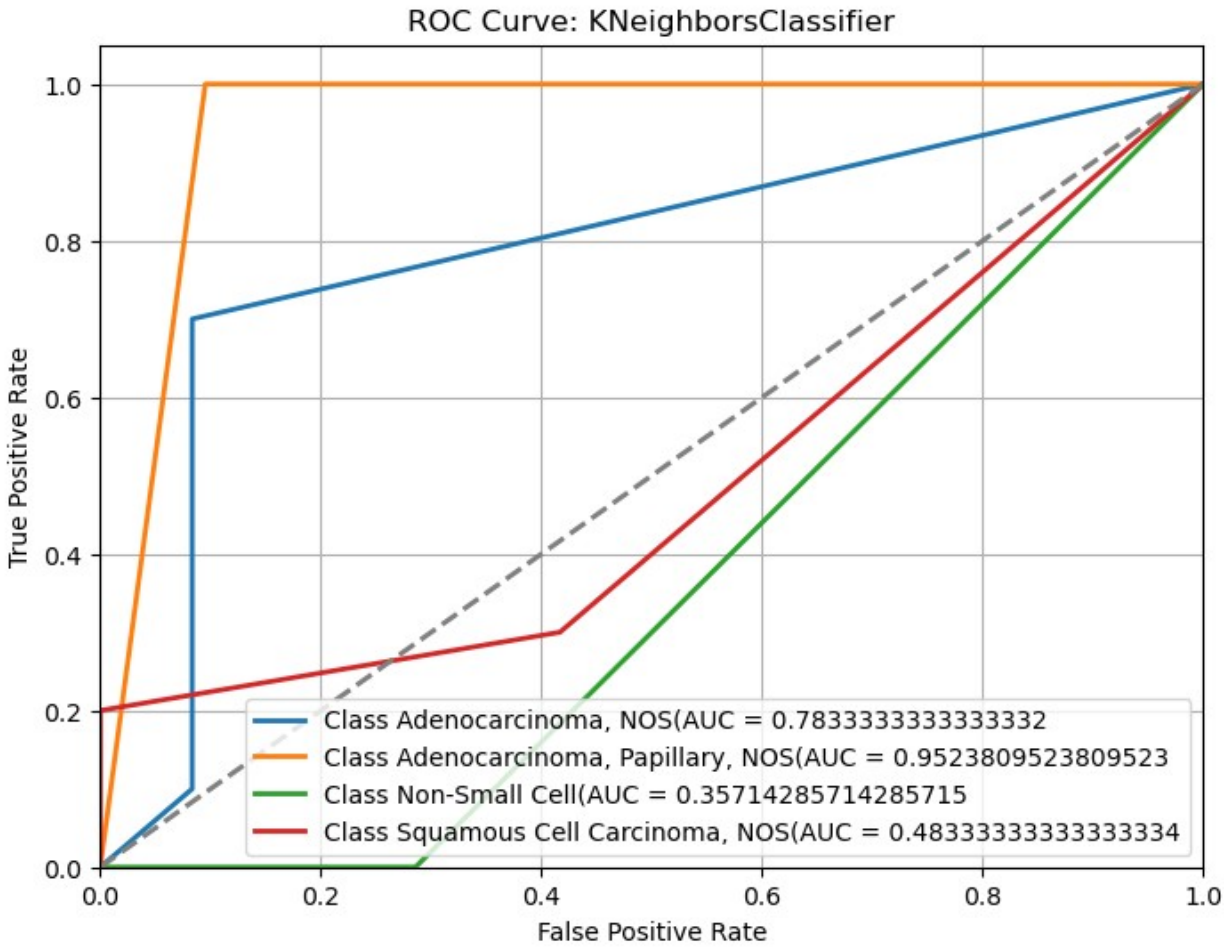
```
Lost X: 15/88 instances.
Lost X: 15/88 instances.



Model name: SVC
Precision: 0.7045454545454546
Accuracy: 0.7727272727272727
Recall: 0.7727272727272727
F1 Score: 0.7355371900826445
```

```
ConfusionMatrix:
 Matrix: [[9 0 0 1]
 [1 0 0 0]
 [0 0 0 1]
 [2 0 0 8]]
Classification report:
Report                                  precision    recall  f1-score
support

           Adenocarcinoma, NOS       0.75      0.90      0.82
10
Adenocarcinoma, Papillary, NOS       0.00      0.00      0.00
1
               Non-Small Cell       0.00      0.00      0.00
1
  Squamous Cell Carcinoma, NOS       0.80      0.80      0.80
10

                     accuracy                           0.77
22
                    macro avg       0.39      0.43      0.40
22
                 weighted avg       0.70      0.77      0.74
22
```



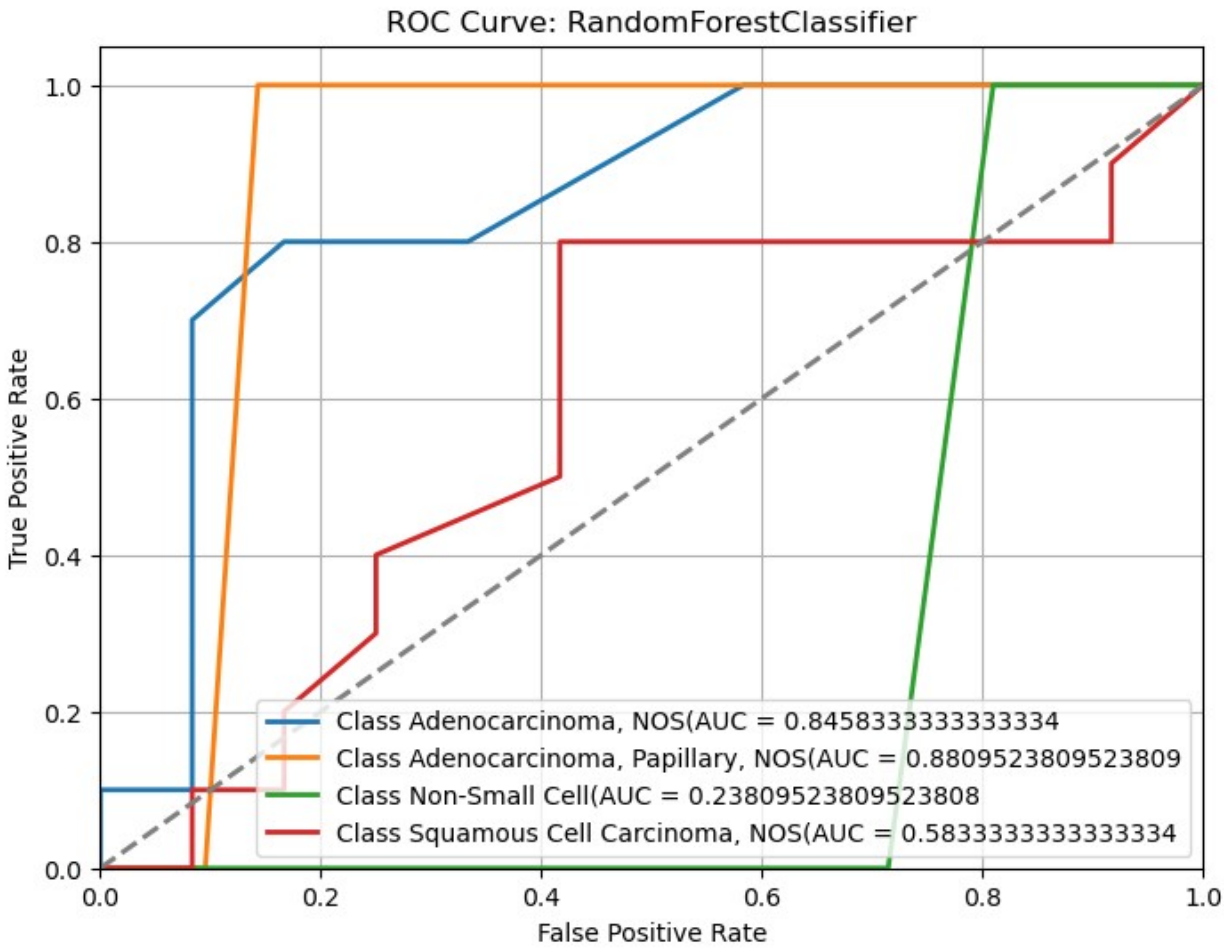

```
Model name: KNeighborsClassifier
Precision: 0.7759412304866852
Accuracy: 0.7727272727272727
Recall: 0.7727272727272727
F1 Score: 0.7723855092276145
ConfusionMatrix:
 Matrix: [[0 0 0 0 0]
 [1 8 0 0 1]
 [0 1 0 0 0]
 [0 0 0 0 1]
 [1 0 0 0 9]]
Classification report:
Report                                                precision
recall  f1-score   support

Adenocarcinoma, Bronchiolo-alveolar Features       0.00      0.00
0.00         0
                         Adenocarcinoma, NOS       0.89      0.80
```

```
0.84         10
  Adenocarcinoma, Papillary, NOS       0.00      0.00
0.00          1
                     Non-Small Cell       0.00      0.00
0.00          1
   Squamous Cell Carcinoma, NOS       0.82      0.90
0.86         10

                              accuracy
0.77         22
                            macro avg       0.34      0.34
0.34         22
                         weighted avg       0.78      0.77
0.77         22
```




```
Model name: RandomForestClassifier
```

```
Precision: 0.7396694214876033
Accuracy: 0.7727272727272727
Recall: 0.7727272727272727
F1 Score: 0.7554112554112554
ConfusionMatrix:
 Matrix: [[8 0 0 2]
 [1 0 0 0]
 [0 0 1 0]
 [2 0 0 8]]
Classification report:
Report                                  precision    recall  f1-score
support

           Adenocarcinoma, NOS       0.73      0.80      0.76
10
Adenocarcinoma, Papillary, NOS       0.00      0.00      0.00
1
                Non-Small Cell       1.00      1.00      1.00
1
  Squamous Cell Carcinoma, NOS       0.80      0.80      0.80
10

                     accuracy                           0.77
22
                    macro avg       0.63      0.65      0.64
22
                 weighted avg       0.74      0.77      0.76
22
```




```
{'SVC': [SVC(kernel='sigmoid', probability=True),
  array(['Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
         'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
         'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
         'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
         'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS'], dtype=object),
  <Figure size 800x600 with 1 Axes>],
 'KNeighborsClassifier': [KNeighborsClassifier(),
  array(['Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
         'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
```

```
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
          'Adenocarcinoma, Bronchiolo-alveolar Features',
          'Squamous Cell Carcinoma, NOS',
          'Adenocarcinoma, Bronchiolo-alveolar Features',
          'Adenocarcinoma, NOS'], dtype=object),
  <Figure size 800x600 with 1 Axes>],
 'RandomForestClassifier': [RandomForestClassifier(random_state=29),
  array(['Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Non-Small Cell', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
          'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS'], dtype=object),
  <Figure size 800x600 with 1 Axes>]}
```

```
from scipy.stats import mode

def vote_soft(model_list, X):
    """
    sums the predic_proba across models and pick the highest. models
must support predict_proba.
    """
    prob_sum = np.sum([model.predict_proba(X) for model in
model_list], axis=0)
    soft_vote_pred = np.argmax(prob_sum, axis=1)
    return soft_vote_pred


def vote_hard(model_list, X):
    #Vote majority across models
    all_preds = np.array([model.predict(X) for model in model_list])
    majority_vote = mode(all_preds, axis=0).mode.flatten()
    return majority_vote

# YOUR CODE HERE TO COMPARE VOTING OUTCOME WITH A BAGGING OR BOOSTING
ALGORITHM FROM SKLEARN
```

```
from VoteClass import Voter
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report

vote = Voter(model_list=models, X_test=eval_class.X_test,
vote_type='s')
results = vote.choose_vote()

boost_model = GradientBoostingClassifier(random_state=42)
boost_model.fit(eval_class.X_train, eval_class.y_train)
boost_pred= boost_model.predict(eval_class.X_test)

print("Boosting Model Results")
print("Accuracy:", accuracy_score(eval_class.y_test, boost_pred))
print("Classification Report:\n",
classification_report(eval_class.y_test, boost_pred, zero_division=0))


print("Vote Model Results")
print("Accuracy:", accuracy_score(eval_class.y_test, results))
print("Classification Report:\n",
classification_report(eval_class.y_test, results, zero_division=0))
```

```
Boosting Model Results
Accuracy: 0.5909090909090909
Classification Report:
                                                    precision    recall
f1-score   support

Adenocarcinoma, Bronchiolo-alveolar Features       0.00      0.00
0.00         0
                          Adenocarcinoma, NOS       0.57      0.40
0.47        10
              Adenocarcinoma, Papillary, NOS       0.00      0.00
0.00         1
                               Non-Small Cell       1.00      1.00
1.00         1
                  Squamous Cell Carcinoma, NOS       1.00      0.80
0.89        10

                                     accuracy
0.59        22
                                    macro avg       0.51      0.44
0.47        22
                                 weighted avg       0.76      0.59
0.66        22

Vote Model Results
Accuracy: 0.7727272727272727
Classification Report:
```

```
                                precision    recall  f1-score
support

           Adenocarcinoma, NOS       0.80      0.80      0.80
10
Adenocarcinoma, Papillary, NOS       0.00      0.00      0.00
1
                Non-Small Cell       0.00      0.00      0.00
1
  Squamous Cell Carcinoma, NOS       0.75      0.90      0.82
10

                      accuracy                           0.77
22
                     macro avg       0.39      0.43      0.40
22
                  weighted avg       0.70      0.77      0.74
22
```

Part E. Pipelines

The sklearn pipeline function is a tool in machine learning that simplifies the workflow by encapsulating all the steps involved in a single object. It offers advantages such as simplicity, reproducibility, efficiency, flexibility, and integration. The Pipeline is built using a list of (key, value) pairs, where the key is a string containing the name you want to give this step and value is an estimator object (the method to be executed).

Pipeline Task

- Read the Study Case notebook for a pipeline functions to understand the principle of the pipeline function
- Implement a pipeline which prepares and classifies data
- Use a `GridSearchCV` object with the `Pipeline` object and a parameter grid to optimize choose the best hyper parameters

```
from DataProcessor import DataProcessor

pipeline = DataProcessor(input_file_exp =
'GSE58661_series_matrix.txt.gz', input_file_meta =
"Lung3.metadata.xls",
                         cols_to_grab = ['title',
       'characteristics.tag.gender', 'characteristics.tag.histology',
       'characteristics.tag.tumor.size.maximumdiameter'],
target='characteristics.tag.histology')

pipeline_finished, encoders = pipeline.pipeline_run()
print(encoders)
```

```
print(pipeline_finished.head())
print(pipeline.look_at_encode(encoders, "title"))
```

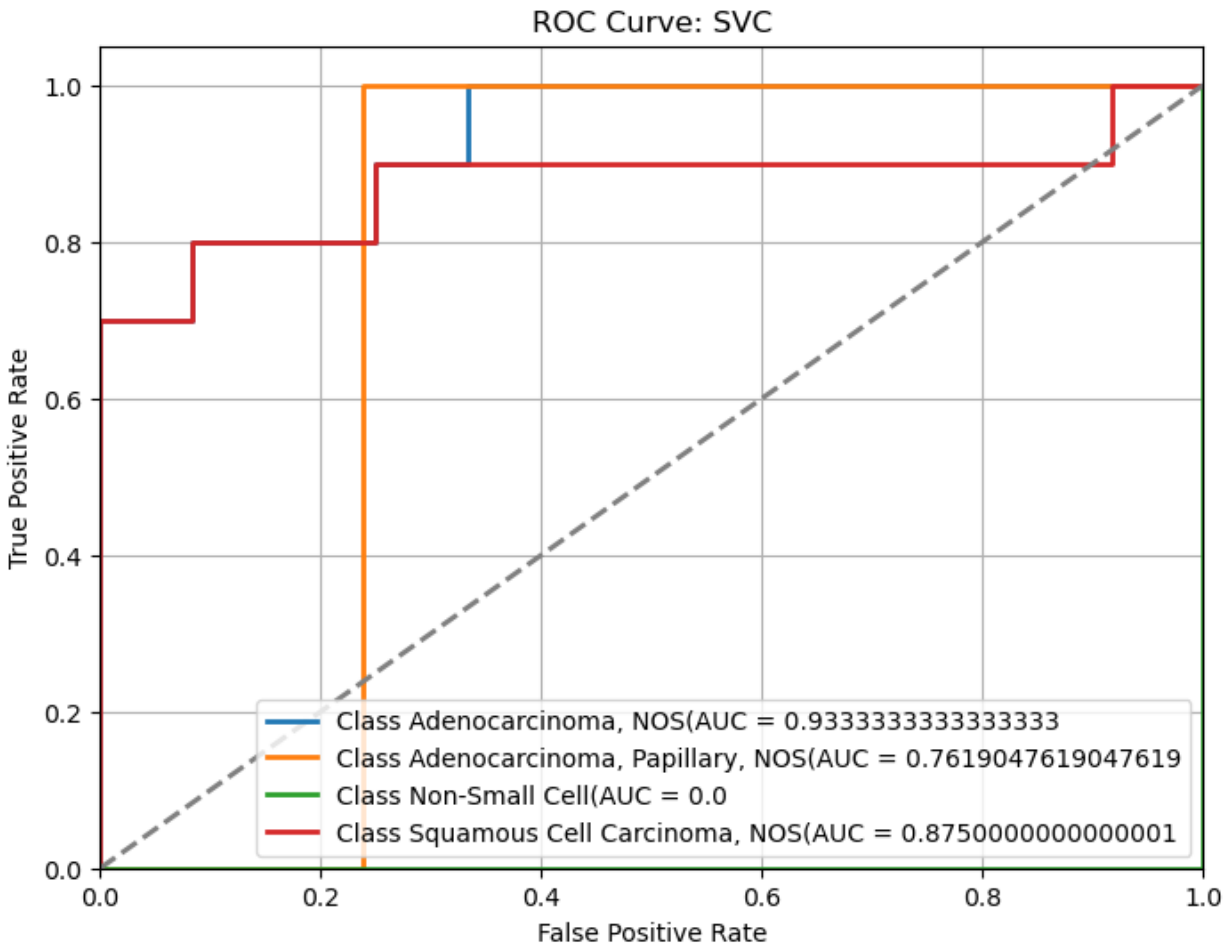
```
N component to reach 50% explained of the variance: 14
Explained variance list: [0.13625197 0.0657839  0.0472555  0.04508168
0.03637887 0.02773954
 0.02481434 0.0213042  0.02095335 0.0170285  0.01591754 0.01557377
 0.01436692 0.01387649]
['title', 'gender', 'histology']
{'title': LabelEncoder(), 'gender': LabelEncoder()}
   title  gender                        histology
tumor.size.maximumdiameter  \
0      0       1    Squamous Cell Carcinoma, NOS
4.0
1     11       1  Adenocarcinoma, Papillary, NOS
1.3
2     22       1                  Non-Small Cell
11.0
4     43       0    Squamous Cell Carcinoma, NOS
7.8
5     54       1             Adenocarcinoma, NOS
3.5

       PCA_1     PCA_2     PCA_3      PCA_4     PCA_5     PCA_6
PCA_7  \
0  15.075265 -5.472475  0.144114   2.997496  3.194623 -0.863421 -
8.363304
1  -6.797404  9.108340 -4.146173  10.091751 -3.858601  2.189105 -
0.724275
2   7.656460  5.641157  2.396656   0.507307 -0.120188  7.554593 -
9.346953
4   6.930503 -1.981333 -3.090020  -2.861382 -4.963344  0.083323
3.793841
5  -6.866965  0.155863 -4.951329  -4.215494  0.506148 -1.350626
0.538385

      PCA_8     PCA_9    PCA_10    PCA_11    PCA_12     PCA_13
PCA_14
0  2.396658  0.470345 -2.003722 -3.817031  0.566312  -2.601190 -
1.235243
1  2.863347 -2.221778 -1.284008  5.740897 -0.271739   2.137453 -
2.982435
2  4.866749 -4.272516  5.607741 -9.701937 -2.217542  11.334382
9.983083
4 -4.423526  4.856949  1.907469  1.411168 -3.258795   2.172437
4.477996
5 -2.566367 -0.201065  0.043919 -2.282812  2.594097   4.914067
0.438946
```

```
{'lung_1': 0, 'lung_10': 1, 'lung_11': 2, 'lung_12': 3, 'lung_13': 4,
'lung_14': 5, 'lung_15': 6, 'lung_16': 7, 'lung_17': 8, 'lung_18': 9,
'lung_19': 10, 'lung_2': 11, 'lung_20': 12, 'lung_21': 13, 'lung_22':
14, 'lung_23': 15, 'lung_24': 16, 'lung_25': 17, 'lung_26': 18,
'lung_27': 19, 'lung_28': 20, 'lung_29': 21, 'lung_3': 22, 'lung_30':
23, 'lung_31': 24, 'lung_32': 25, 'lung_33': 26, 'lung_34': 27,
'lung_35': 28, 'lung_36': 29, 'lung_37': 30, 'lung_38': 31, 'lung_39':
32, 'lung_40': 33, 'lung_41': 34, 'lung_42': 35, 'lung_43': 36,
'lung_44': 37, 'lung_45': 38, 'lung_46': 39, 'lung_47': 40, 'lung_48':
41, 'lung_49': 42, 'lung_5': 43, 'lung_50': 44, 'lung_51': 45,
'lung_52': 46, 'lung_53': 47, 'lung_54': 48, 'lung_55': 49, 'lung_56':
50, 'lung_57': 51, 'lung_58': 52, 'lung_59': 53, 'lung_6': 54,
'lung_60': 55, 'lung_61': 56, 'lung_62': 57, 'lung_63': 58, 'lung_64':
59, 'lung_65': 60, 'lung_66': 61, 'lung_67': 62, 'lung_68': 63,
'lung_69': 64, 'lung_7': 65, 'lung_70': 66, 'lung_71': 67, 'lung_72':
68, 'lung_73': 69, 'lung_74': 70, 'lung_75': 71, 'lung_76': 72,
'lung_77': 73, 'lung_78': 74, 'lung_79': 75, 'lung_8': 76, 'lung_80':
77, 'lung_81': 78, 'lung_82': 79, 'lung_83': 80, 'lung_84': 81,
'lung_85': 82, 'lung_86': 83, 'lung_87': 84, 'lung_88': 85, 'lung_89':
86, 'lung_9': 87}
{'lung_1': 0, 'lung_10': 1, 'lung_11': 2, 'lung_12': 3, 'lung_13': 4,
'lung_14': 5, 'lung_15': 6, 'lung_16': 7, 'lung_17': 8, 'lung_18': 9,
'lung_19': 10, 'lung_2': 11, 'lung_20': 12, 'lung_21': 13, 'lung_22':
14, 'lung_23': 15, 'lung_24': 16, 'lung_25': 17, 'lung_26': 18,
'lung_27': 19, 'lung_28': 20, 'lung_29': 21, 'lung_3': 22, 'lung_30':
23, 'lung_31': 24, 'lung_32': 25, 'lung_33': 26, 'lung_34': 27,
'lung_35': 28, 'lung_36': 29, 'lung_37': 30, 'lung_38': 31, 'lung_39':
32, 'lung_40': 33, 'lung_41': 34, 'lung_42': 35, 'lung_43': 36,
'lung_44': 37, 'lung_45': 38, 'lung_46': 39, 'lung_47': 40, 'lung_48':
41, 'lung_49': 42, 'lung_5': 43, 'lung_50': 44, 'lung_51': 45,
'lung_52': 46, 'lung_53': 47, 'lung_54': 48, 'lung_55': 49, 'lung_56':
50, 'lung_57': 51, 'lung_58': 52, 'lung_59': 53, 'lung_6': 54,
'lung_60': 55, 'lung_61': 56, 'lung_62': 57, 'lung_63': 58, 'lung_64':
59, 'lung_65': 60, 'lung_66': 61, 'lung_67': 62, 'lung_68': 63,
'lung_69': 64, 'lung_7': 65, 'lung_70': 66, 'lung_71': 67, 'lung_72':
68, 'lung_73': 69, 'lung_74': 70, 'lung_75': 71, 'lung_76': 72,
'lung_77': 73, 'lung_78': 74, 'lung_79': 75, 'lung_8': 76, 'lung_80':
77, 'lung_81': 78, 'lung_82': 79, 'lung_83': 80, 'lung_84': 81,
'lung_85': 82, 'lung_86': 83, 'lung_87': 84, 'lung_88': 85, 'lung_89':
86, 'lung_9': 87}
```

```
from Evaluation import Evaluator
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

svccl = SVC(kernel='sigmoid', probability=True)
kncl = KNeighborsClassifier(n_neighbors=5)
rfcl = RandomForestClassifier(random_state=29)
```

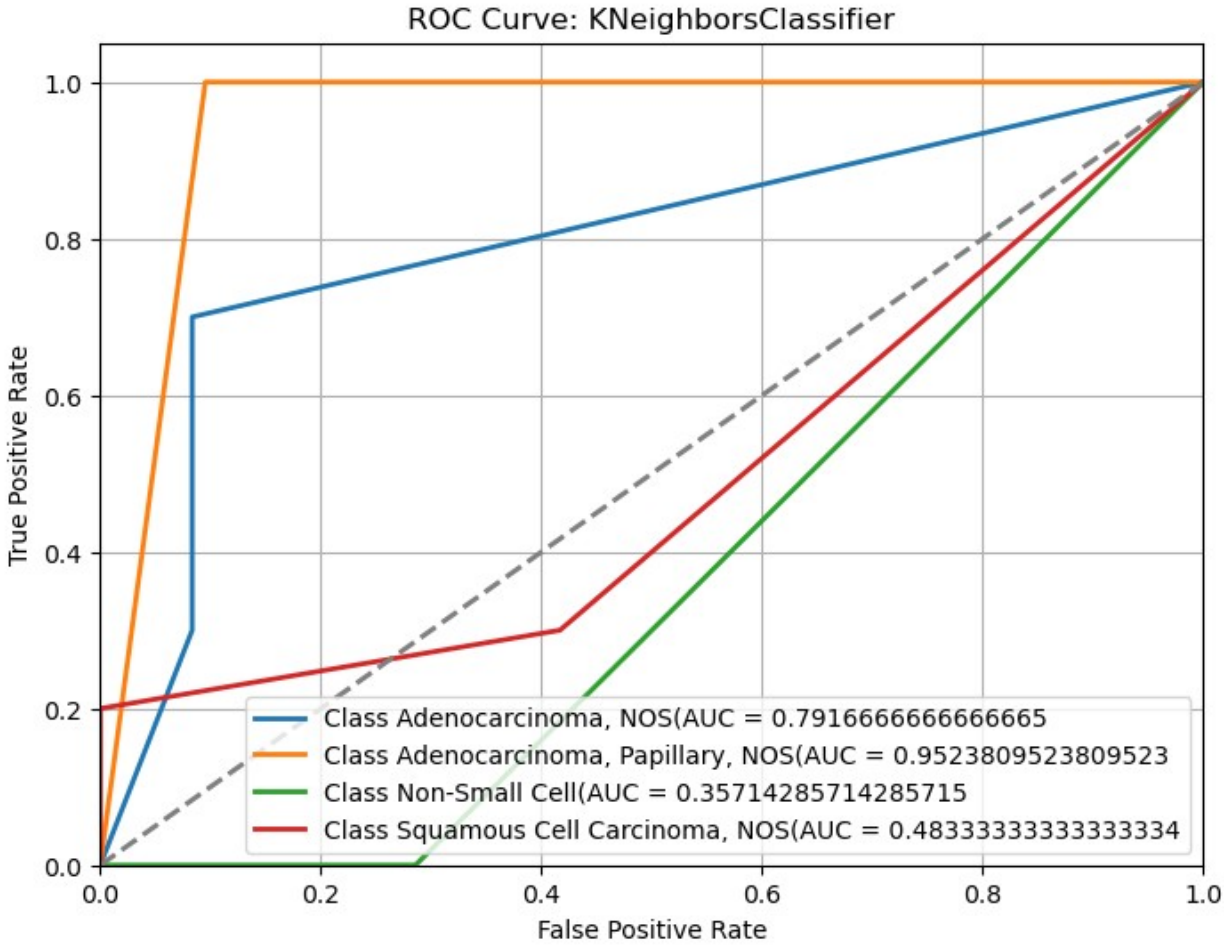
```
models = [svccl, knncl, rfcl]

y = pipeline_finished['histology']
X = pipeline_finished.drop(columns=['histology', 'title', 'gender'])

eval_class = Evaluator(models, X, y)

eval_class.execute()
```

```
Lost X: 15/88 instances.
Lost X: 15/88 instances.



Model name: SVC
Precision: 0.7045454545454546
Accuracy: 0.7727272727272727
Recall: 0.7727272727272727
F1 Score: 0.7355371900826445
ConfusionMatrix:
 Matrix: [[9 0 0 1]
 [1 0 0 0]
 [0 0 0 1]
 [2 0 0 8]]
Classification report:
Report                                  precision    recall  f1-score
support

           Adenocarcinoma, NOS       0.75      0.90      0.82
10
Adenocarcinoma, Papillary, NOS       0.00      0.00      0.00
1
                Non-Small Cell       0.00      0.00      0.00
1
  Squamous Cell Carcinoma, NOS       0.80      0.80      0.80
10

                     accuracy                           0.77
22
                    macro avg       0.39      0.43      0.40
22
                 weighted avg       0.70      0.77      0.74
22
```



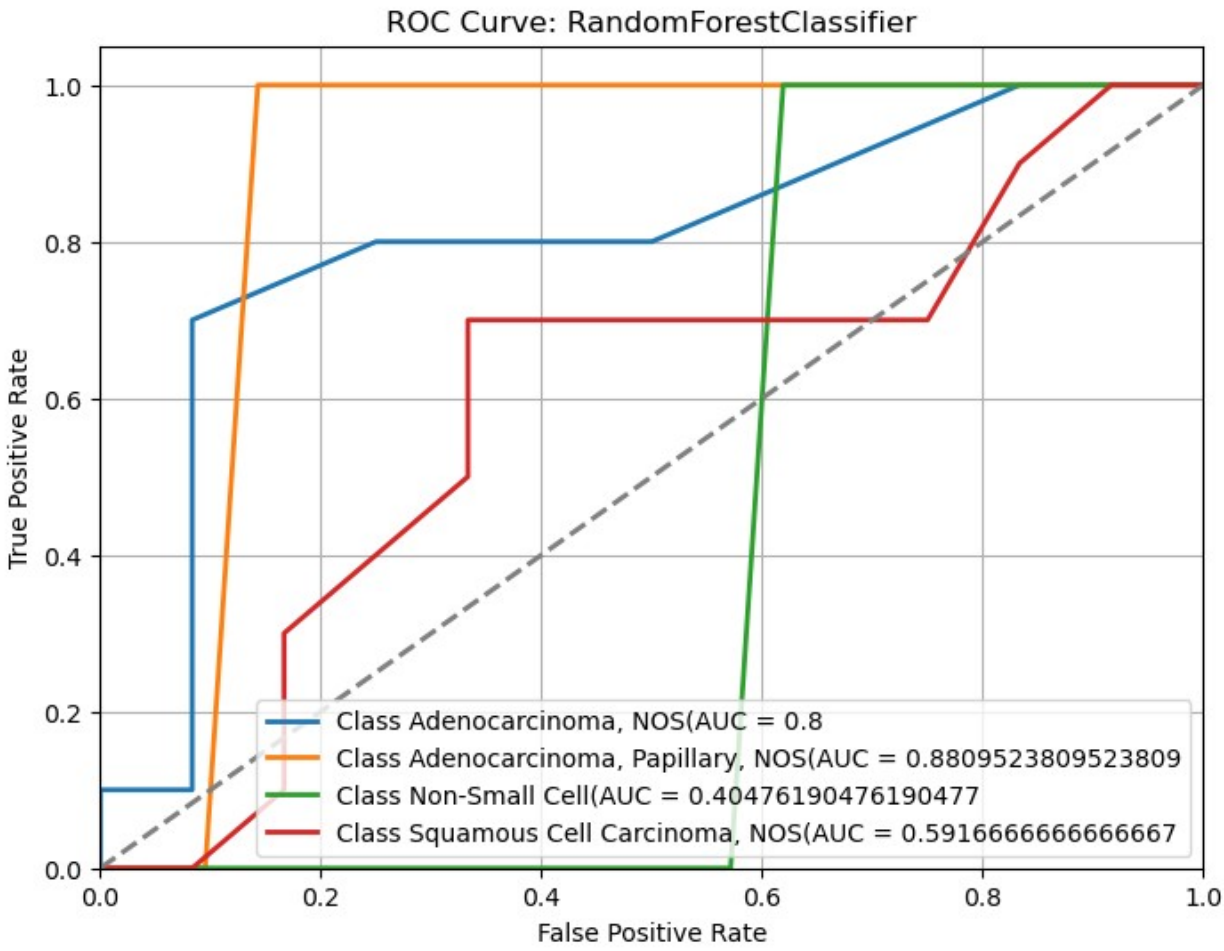

```
Model name: KNeighborsClassifier
Precision: 0.7196969696969697
Accuracy: 0.6363636363636364
Recall: 0.6363636363636364
F1 Score: 0.65599173553719
ConfusionMatrix:
 Matrix: [[0 0 0 0 0]
 [3 5 0 0 2]
 [0 1 0 0 0]
 [0 0 0 0 1]
 [1 0 0 0 9]]
Classification report:
Report                                                precision
recall  f1-score   support

Adenocarcinoma, Bronchiolo-alveolar Features       0.00      0.00
0.00         0
                         Adenocarcinoma, NOS       0.83      0.50
```

```
0.62         10
  Adenocarcinoma, Papillary, NOS       0.00      0.00
0.00          1
                     Non-Small Cell       0.00      0.00
0.00          1
   Squamous Cell Carcinoma, NOS       0.75      0.90
0.82         10

                              accuracy
0.64         22
                            macro avg       0.32      0.28
0.29         22
                         weighted avg       0.72      0.64
0.66         22
```




```
Model name: RandomForestClassifier
```

```
Precision: 0.6942148760330579
Accuracy: 0.7272727272727273
Recall: 0.7272727272727273
F1 Score: 0.7099567099567099
ConfusionMatrix:
 Matrix: [[7 0 0 3]
 [1 0 0 0]
 [0 0 1 0]
 [2 0 0 8]]
Classification report:
Report                                  precision    recall  f1-score
support

           Adenocarcinoma, NOS       0.70      0.70      0.70
10
Adenocarcinoma, Papillary, NOS       0.00      0.00      0.00
1
                Non-Small Cell       1.00      1.00      1.00
1
  Squamous Cell Carcinoma, NOS       0.73      0.80      0.76
10

                     accuracy                           0.73
22
                    macro avg       0.61      0.62      0.62
22
                 weighted avg       0.69      0.73      0.71
22
```




```
{'SVC': [SVC(kernel='sigmoid', probability=True),
  array(['Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
         'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
         'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
         'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
         'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS'], dtype=object),
  <Figure size 800x600 with 1 Axes>],
 'KNeighborsClassifier': [KNeighborsClassifier(),
  array(['Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
         'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
         'Squamous Cell Carcinoma, NOS',
```

```
          'Adenocarcinoma, Bronchiolo-alveolar Features',
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
          'Adenocarcinoma, Bronchiolo-alveolar Features',
          'Squamous Cell Carcinoma, NOS',
          'Adenocarcinoma, Bronchiolo-alveolar Features',
          'Adenocarcinoma, Bronchiolo-alveolar Features'],
dtype=object),
  <Figure size 800x600 with 1 Axes>],
 'RandomForestClassifier': [RandomForestClassifier(random_state=29),
  array(['Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Non-Small Cell', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Adenocarcinoma, NOS',
          'Adenocarcinoma, NOS', 'Squamous Cell Carcinoma, NOS',
          'Squamous Cell Carcinoma, NOS', 'Squamous Cell Carcinoma,
NOS',
          'Adenocarcinoma, NOS', 'Adenocarcinoma, NOS'], dtype=object),
  <Figure size 800x600 with 1 Axes>]}
```

```
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from Evaluation import Evaluator
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from VoteClass import Voter
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report

svccl = SVC(kernel='sigmoid', probability=True)
knncl = KNeighborsClassifier(n_neighbors=5)
rfcl = RandomForestClassifier(random_state=29)

models = [svccl, knncl, rfcl]
```

```
y = pipeline_finished['histology'].copy()
X = pipeline_finished.drop(columns=['histology', 'title', 'gender'])

rare_classes = y.value_counts()[y.value_counts() < 4].index  # less
than 3 samples
y = y.replace(rare_classes, 'Other')



# Find minimum samples in a class
min_class_samples = y.value_counts().min()
print("Minimum samples in a class:", min_class_samples)

# Use this for cv
cv_strategy = StratifiedKFold(n_splits=min(min_class_samples, 5))

eval_class = Evaluator(models, X, y)
eval_class.get_train_test_split()

def pipeline_acceptor(model_pipeline, model_param_grid, cv_strategy,
eval_class):
    model_grid = GridSearchCV(model_pipeline, model_param_grid,
cv=cv_strategy, scoring='accuracy')
    model_grid.fit(eval_class.X_train, eval_class.y_train)
    model_name = model_pipeline.steps[-1][0].upper()
    print(f"Best {model_name} params:", model_grid.best_params_)
    print(f"Best {model_name} score:", model_grid.best_score_)
    return model_grid.best_estimator_



# Pipeline + GridSearchCV for SVC
svc_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svc', SVC(probability=True))
])

svc_param_grid = {
    'svc__kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'svc__C': [0.1, 1, 10, 100],
    'svc__gamma': ['scale', 'auto']
}


# Pipeline + GridSearchCV for KNN
knn_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('knn', KNeighborsClassifier())
])
```

```
knn_param_grid = {
    'knn__n_neighbors': [3, 5, 7, 9],
    'knn__weights': ['uniform', 'distance'],
    'knn__p': [1, 2]  # 1=Manhattan, 2=Euclidean
}

# Pipeline + GridSearchCV for Random Forest
rf_pipeline = Pipeline([
    ('rf', RandomForestClassifier(random_state=29))
])

rf_param_grid = {
    'rf__n_estimators': [50, 100, 200],
    'rf__max_depth': [None, 5, 10, 20],
    'rf__min_samples_split': [2, 5, 10],
    'rf__min_samples_leaf': [1, 2, 4]
}



model_list2 = [pipeline_acceptor(rf_pipeline, rf_param_grid,
cv_strategy, eval_class), pipeline_acceptor(knn_pipeline,
knn_param_grid, cv_strategy, eval_class),
pipeline_acceptor(svc_pipeline, svc_param_grid, cv_strategy,
eval_class)]

vote = Voter(model_list=model_list2, X_test=eval_class.X_test,
vote_type='s')
results = vote.choose_vote()

print("Vote Model Results")
print("Accuracy:", accuracy_score(eval_class.y_test, results))
print("Classification Report:\n",
classification_report(eval_class.y_test, results, zero_division=0))
```

```
Minimum samples in a class: 4

/home/jippe/anaconda3/envs/dsc5/lib/python3.13/site-packages/sklearn/
model_selection/_split.py:811: UserWarning: The least populated class
in y has only 1 members, which is less than n_splits=4.
  warnings.warn(

Best RF params: {'rf__max_depth': 5, 'rf__min_samples_leaf': 1,
'rf__min_samples_split': 5, 'rf__n_estimators': 50}
Best RF score: 0.6229166666666667

/home/jippe/anaconda3/envs/dsc5/lib/python3.13/site-packages/sklearn/
model_selection/_split.py:811: UserWarning: The least populated class
in y has only 1 members, which is less than n_splits=4.
  warnings.warn(
```

```
Best KNN params: {'knn__n_neighbors': 5, 'knn__p': 1, 'knn__weights':
'distance'}
Best KNN score: 0.6229166666666668

/home/jippe/anaconda3/envs/dsc5/lib/python3.13/site-packages/sklearn/
model_selection/_split.py:811: UserWarning: The least populated class
in y has only 1 members, which is less than n_splits=4.
  warnings.warn(

Best SVC params: {'svc__C': 10, 'svc__gamma': 'scale', 'svc__kernel':
'rbf'}
Best SVC score: 0.6541666666666667
Vote Model Results
Accuracy: 0.37037037037037035
Classification Report:
                                                precision    recall
f1-score   support

Adenocarcinoma, Bronchiolo-alveolar Features       0.00      0.00
0.00         3
                         Adenocarcinoma, NOS       0.40      0.25
0.31         8
             Adenocarcinoma, Papillary, NOS       0.00      0.00
0.00         3
                           Non-Small Cell       0.00      0.00
0.00         3
                                    Other       0.20      0.33
0.25         3
               Squamous Cell Carcinoma, NOS       0.41      1.00
0.58         7

                                 accuracy
0.37        27
                                macro avg       0.17      0.26
0.19        27
                             weighted avg       0.25      0.37
0.27        27


from PipelineOptimize import PipelineMaker

y = pipeline_finished['histology'].copy()
X = pipeline_finished.drop(columns=['histology', 'title', 'gender'])

# Pipeline + GridSearchCV for SVC
svc_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svc', SVC(probability=True))
])

svc_param_grid = {
```

```
    'svc__kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'svc__C': [0.1, 1, 10, 100],
    'svc__gamma': ['scale', 'auto']
}

# Pipeline + GridSearchCV for KNN
knn_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('knn', KNeighborsClassifier())
])

knn_param_grid = {
    'knn__n_neighbors': [3, 5, 7, 9],
    'knn__weights': ['uniform', 'distance'],
    'knn__p': [1, 2]  # 1=Manhattan, 2=Euclidean
}

# Pipeline + GridSearchCV for Random Forest
rf_pipeline = Pipeline([
    ('rf', RandomForestClassifier(random_state=29))
])

rf_param_grid = {
    'rf__n_estimators': [50, 100, 200],
    'rf__max_depth': [None, 5, 10, 20],
    'rf__min_samples_split': [2, 5, 10],
    'rf__min_samples_leaf': [1, 2, 4]
}


pipeline_grid_list = [[svc_pipeline, svc_param_grid], [rf_pipeline,
rf_param_grid], [knn_pipeline, knn_param_grid]]

piper = PipelineMaker(X=X,y=y, pipeline_grid_list=pipeline_grid_list)
best_model_list = piper.execute_all()

print(best_model_list)
```

```
/home/jippe/anaconda3/envs/dsc5/lib/python3.13/site-packages/sklearn/
model_selection/_split.py:811: UserWarning: The least populated class
in y has only 1 members, which is less than n_splits=4.
  warnings.warn(

Best SVC params: {'svc__C': 10, 'svc__gamma': 'scale', 'svc__kernel':
'rbf'}
Best SVC score: 0.6541666666666667

/home/jippe/anaconda3/envs/dsc5/lib/python3.13/site-packages/sklearn/
model_selection/_split.py:811: UserWarning: The least populated class
in y has only 1 members, which is less than n_splits=4.
  warnings.warn(
```

```
Best RF params: {'rf__max_depth': 5, 'rf__min_samples_leaf': 1,
'rf__min_samples_split': 5, 'rf__n_estimators': 50}
Best RF score: 0.6229166666666667

/home/jippe/anaconda3/envs/dsc5/lib/python3.13/site-packages/sklearn/
model_selection/_split.py:811: UserWarning: The least populated class
in y has only 1 members, which is less than n_splits=4.
  warnings.warn(

Best KNN params: {'knn__n_neighbors': 5, 'knn__p': 1, 'knn__weights':
'distance'}
Best KNN score: 0.6229166666666668
[Pipeline(steps=[('scaler', StandardScaler()),
                 ('svc', SVC(C=10, probability=True))]),
Pipeline(steps=[('rf',
                  RandomForestClassifier(max_depth=5,
min_samples_split=5,
                                         n_estimators=50,
random_state=29))]), Pipeline(steps=[('scaler', StandardScaler()),
                 ('knn', KNeighborsClassifier(p=1,
weights='distance'))])]

piper_solo = PipelineMaker(X=X, y=y, model_pipeline=svc_pipeline,
model_param_grid=svc_param_grid)
best_model = piper_solo.execute_all()
print(best_model)

/home/jippe/anaconda3/envs/dsc5/lib/python3.13/site-packages/sklearn/
model_selection/_split.py:811: UserWarning: The least populated class
in y has only 1 members, which is less than n_splits=4.
  warnings.warn(

Best SVC params: {'svc__C': 10, 'svc__gamma': 'scale', 'svc__kernel':
'rbf'}
Best SVC score: 0.6541666666666667
Pipeline(steps=[('scaler', StandardScaler()),
                ('svc', SVC(C=10, probability=True))])
```

Bring it all together

By now, you've developed code snippets for model evaluation, optimization, and data improvement. Now, leverage these skills to build a classification model using the Clinical and Genetic Lung data. Make sure that you log your experiments.

Once you're satisfied with the model, upload the relevant code to your repository and or refactor code with new insights. Furthermore, take a moment to reflect on its applicability and potential real-world impact. Update your evaluation document(s) with these findings in your repository.

Bonus: ML for operations

If we intend to deploy the model in a real-world application, it's more efficient to save and reuse the trained model rather than retraining it each time.

- Read the blog: https://neptune.ai/blog/saving-trained-model-in-python
- Write three python files 1) a train_model python file 2) a use_model python file 3) a retrain_model python file that adds new data to the original training data and updates the model
- Update your repository