Molecular Classification of Cancer by Gene Expression Monitoring

In [1]:
```
# Edit all the Mardown cells below with the appropriate information
# Run all cells, containing your code
# Save this Jupyter with the outputs of your executed cells
# PS: Save again the notebook with this outcome.
# PSPS: Don't forget to include the dataset in your submission
```

Team:

- Anthony Lam

Course: CISB 60 – ML and DL (Fall, 2024)

Problem Statement

- This project is about trying to classify two different types of cancer (leukemia) from Gene Expression Data. The dataset can be found at:

https://www.kaggle.com/datasets/crawford/gene-expression/data?select=data_set_ALL_AML_train.csv

- **Keywords:** Acute Myeloid Leukemia, Acute Lymphoblastic Leukemia, Gene Expression

Required packages

- Add instructions to install the required packages

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
#Load dataset
train_df = pd.read_csv("data/data_set_ALL_AML_train.csv")
test_df = pd.read_csv("data/data_set_ALL_AML_independent.csv")
actual_df = pd.read_csv("data/actual.csv", index_col = 'patient')
```

In [4]:
```
train_df.head()
```

Out[4]:

	Gene Description	Gene Accession Number	1	call	2	call.1	3	call.2	4	call.3	...	29	call.33	3(
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-214	A	-139	A	-76	A	-135	A	...	15	A	-31
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-153	A	-73	A	-49	A	-114	A	...	-114	A	-19
2	AFFX-BioB-3_at (endogenous control)	AFFX-BioB-3_at	-58	A	-1	A	-307	A	265	A	...	2	A	-9
3	AFFX-BioC-5_at (endogenous control)	AFFX-BioC-5_at	88	A	283	A	309	A	12	A	...	193	A	31
4	AFFX-BioC-3_at (endogenous control)	AFFX-BioC-3_at	-295	A	-264	A	-376	A	-419	A	...	-51	A	-13

5 rows × 78 columns

In [5]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7129 entries, 0 to 7128
Data columns (total 78 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Gene Description       7129 non-null   object
 1   Gene Accession Number  7129 non-null   object
 2   1                      7129 non-null   int64 
 3   call                   7129 non-null   object
 4   2                      7129 non-null   int64 
 5   call.1                 7129 non-null   object
 6   3                      7129 non-null   int64 
 7   call.2                 7129 non-null   object
 8   4                      7129 non-null   int64 
 9   call.3                 7129 non-null   object
 10  5                      7129 non-null   int64 
 11  call.4                 7129 non-null   object
 12  6                      7129 non-null   int64 
 13  call.5                 7129 non-null   object
 14  7                      7129 non-null   int64 
 15  call.6                 7129 non-null   object
 16  8                      7129 non-null   int64 
 17  call.7                 7129 non-null   object
 18  9                      7129 non-null   int64 
 19  call.8                 7129 non-null   object
 20  10                     7129 non-null   int64 
 21  call.9                 7129 non-null   object
 22  11                     7129 non-null   int64 
 23  call.10                7129 non-null   object
 24  12                     7129 non-null   int64 
 25  call.11                7129 non-null   object
 26  13                     7129 non-null   int64 
 27  call.12                7129 non-null   object
 28  14                     7129 non-null   int64 
 29  call.13                7129 non-null   object
 30  15                     7129 non-null   int64 
 31  call.14                7129 non-null   object
 32  16                     7129 non-null   int64 
 33  call.15                7129 non-null   object
 34  17                     7129 non-null   int64 
 35  call.16                7129 non-null   object
 36  18                     7129 non-null   int64 
 37  call.17                7129 non-null   object
 38  19                     7129 non-null   int64 
 39  call.18                7129 non-null   object
 40  20                     7129 non-null   int64 
 41  call.19                7129 non-null   object
 42  21                     7129 non-null   int64 
 43  call.20                7129 non-null   object
 44  22                     7129 non-null   int64 
 45  call.21                7129 non-null   object
 46  23                     7129 non-null   int64 
 47  call.22                7129 non-null   object
 48  24                     7129 non-null   int64 
 49  call.23                7129 non-null   object
 50  25                     7129 non-null   int64 
 51  call.24                7129 non-null   object
 52  26                     7129 non-null   int64 
 53  call.25                7129 non-null   object
```

```
 54  27                      7129 non-null   int64
 55  call.26                 7129 non-null   object
 56  34                      7129 non-null   int64
 57  call.27                 7129 non-null   object
 58  35                      7129 non-null   int64
 59  call.28                 7129 non-null   object
 60  36                      7129 non-null   int64
 61  call.29                 7129 non-null   object
 62  37                      7129 non-null   int64
 63  call.30                 7129 non-null   object
 64  38                      7129 non-null   int64
 65  call.31                 7129 non-null   object
 66  28                      7129 non-null   int64
 67  call.32                 7129 non-null   object
 68  29                      7129 non-null   int64
 69  call.33                 7129 non-null   object
 70  30                      7129 non-null   int64
 71  call.34                 7129 non-null   object
 72  31                      7129 non-null   int64
 73  call.35                 7129 non-null   object
 74  32                      7129 non-null   int64
 75  call.36                 7129 non-null   object
 76  33                      7129 non-null   int64
 77  call.37                 7129 non-null   object
dtypes: int64(38), object(40)
memory usage: 4.2+ MB
```

In [12]:
```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7129 entries, 0 to 7128
Data columns (total 70 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Gene Description       7129 non-null   object
 1   Gene Accession Number  7129 non-null   object
 2   39                     7129 non-null   int64 
 3   call                   7129 non-null   object
 4   40                     7129 non-null   int64 
 5   call.1                 7129 non-null   object
 6   42                     7129 non-null   int64 
 7   call.2                 7129 non-null   object
 8   47                     7129 non-null   int64 
 9   call.3                 7129 non-null   object
 10  48                     7129 non-null   int64 
 11  call.4                 7129 non-null   object
 12  49                     7129 non-null   int64 
 13  call.5                 7129 non-null   object
 14  41                     7129 non-null   int64 
 15  call.6                 7129 non-null   object
 16  43                     7129 non-null   int64 
 17  call.7                 7129 non-null   object
 18  44                     7129 non-null   int64 
 19  call.8                 7129 non-null   object
 20  45                     7129 non-null   int64 
 21  call.9                 7129 non-null   object
 22  46                     7129 non-null   int64 
 23  call.10                7129 non-null   object
 24  70                     7129 non-null   int64 
 25  call.11                7129 non-null   object
 26  71                     7129 non-null   int64 
 27  call.12                7129 non-null   object
 28  72                     7129 non-null   int64 
 29  call.13                7129 non-null   object
 30  68                     7129 non-null   int64 
 31  call.14                7129 non-null   object
 32  69                     7129 non-null   int64 
 33  call.15                7129 non-null   object
 34  67                     7129 non-null   int64 
 35  call.16                7129 non-null   object
 36  55                     7129 non-null   int64 
 37  call.17                7129 non-null   object
 38  56                     7129 non-null   int64 
 39  call.18                7129 non-null   object
 40  59                     7129 non-null   int64 
 41  call.19                7129 non-null   object
 42  52                     7129 non-null   int64 
 43  call.20                7129 non-null   object
 44  53                     7129 non-null   int64 
 45  call.21                7129 non-null   object
 46  51                     7129 non-null   int64 
 47  call.22                7129 non-null   object
 48  50                     7129 non-null   int64 
 49  call.23                7129 non-null   object
 50  54                     7129 non-null   int64 
 51  call.24                7129 non-null   object
 52  57                     7129 non-null   int64 
 53  call.25                7129 non-null   object
```

```
 54  58                      7129 non-null   int64
 55  call.26                 7129 non-null   object
 56  60                      7129 non-null   int64
 57  call.27                 7129 non-null   object
 58  61                      7129 non-null   int64
 59  call.28                 7129 non-null   object
 60  65                      7129 non-null   int64
 61  call.29                 7129 non-null   object
 62  66                      7129 non-null   int64
 63  call.30                 7129 non-null   object
 64  63                      7129 non-null   int64
 65  call.31                 7129 non-null   object
 66  64                      7129 non-null   int64
 67  call.32                 7129 non-null   object
 68  62                      7129 non-null   int64
 69  call.33                 7129 non-null   object
dtypes: int64(34), object(36)
memory usage: 3.8+ MB
```

In [7]:
```
actual_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 72 entries, 1 to 72
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   cancer  72 non-null     object
dtypes: object(1)
memory usage: 1.1+ KB
```

In [8]:
```
# count patients with which type of cancer
actual_df['cancer'].value_counts()
```

Out[8]:
```
cancer
ALL    47
AML    25
Name: count, dtype: int64
```

In [37]:
```
y = actual_df.replace({'ALL': 0, 'AML': 1})
y
```

Out[37]:

patient	cancer
1	0
2	0
3	0
4	0
5	0
...	...
68	0
69	0
70	0
71	0
72	0

72 rows × 1 columns

In [33]:
```
# remove unneeded columns, and transpose for easier wrangling
train_to_keep=[col for col in train_df.columns if "call" not in col]
test_to_keep=[col for col in test_df.columns if "call" not in col]

X_train = train_df[train_to_keep].T
X_train.columns = X_train.iloc[1]
X_train = X_train.iloc[2:]

X_test = test_df[test_to_keep].T
X_test.columns = X_test.iloc[1]
X_test = X_test.iloc[2:]
```

In [35]:
```
X_train.head()
```

Out[35]:

Gene Accession Number	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at
1	-214	-153	-58	88	-295	-558	199	-176
2	-139	-73	-1	283	-264	-400	-330	-168
3	-76	-49	-307	309	-376	-650	33	-367
4	-135	-114	265	12	-419	-585	158	-253
5	-106	-125	-76	168	-230	-284	4	-122

5 rows × 7129 columns

In [36]:
```
X_test.head()
```

Out[36]:

Gene Accession Number	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at
39	-342	-200	41	328	-224	-427	-656	-292
40	-87	-248	262	295	-226	-493	367	-452
42	22	-153	17	276	-211	-250	55	-141
47	-243	-218	-163	182	-289	-268	-285	-172
48	-130	-177	-28	266	-170	-326	-222	-93

5 rows × 7129 columns

In [39]:
```
# according to dataset doc, first 38 are training, last 34 are test
y_train = y['cancer'][:38]
y_test = y['cancer'][38:]
```

In [41]:
```
# prep feature extraction
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [43]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components = 0.95)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

In [44]:
```
X_train_pca.shape
```

Out[44]:
```
(38, 32)
```

In [45]:
```
X_test_pca.shape
```

Out[45]:
```
(34, 32)
```

95% of variance explained in 32 components

Methodology

1. Explan your ML and DL metodology

1. Introduce the topics you used in your project

    - Model 1
        - Naive Bayes
    - Model 2
        - Deep Learning

Your code starts here

Machine Learning Section

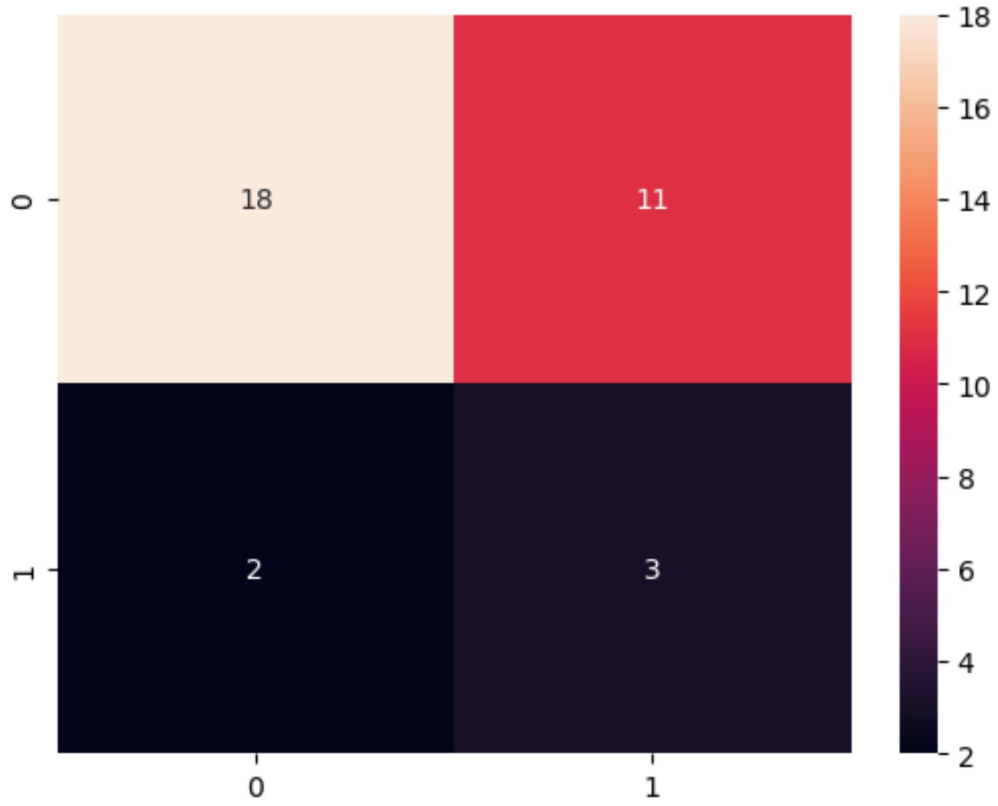
In [46]:
```
from sklearn.naive_bayes import GaussianNB

ml_model = GaussianNB()

ml_model.fit(X_train_pca, y_train)
```

Out[46]:
```
▾  GaussianNB i ?

GaussianNB()
```

In [53]:
```
from sklearn.metrics import classification_report, confusion_matrix

ml_prediction = ml_model.predict(X_test_pca)
ml_accuracy = accuracy_score(ml_prediction, y_test)

print(ml_accuracy)
print(classification_report(ml_prediction, y_test))

sns.heatmap(confusion_matrix(ml_prediction, y_test), annot=True)
```

```
0.6176470588235294
              precision    recall  f1-score   support

           0       0.90      0.62      0.73        29
           1       0.21      0.60      0.32         5

    accuracy                           0.62        34
   macro avg       0.56      0.61      0.53        34
weighted avg       0.80      0.62      0.67        34
```

Out[53]:
```
<Axes: >
```



Deep Learning Section

In [94]:
```
import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.optimizers import SGD

dl_model = Sequential()
dl_model.add(layers.Dense(32, activation='relu', input_shape=X_train_pca[1].shape))
dl_model.add(layers.Dense(16, activation='relu'))
dl_model.add(layers.Dense(1, activation='sigmoid'))
```

In [95]:
```
dl_model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_21 (Dense)            (None, 32)                1056

 dense_22 (Dense)            (None, 16)                528

 dense_23 (Dense)            (None, 1)                 17

=================================================================
Total params: 1601 (6.25 KB)
Trainable params: 1601 (6.25 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [96]:
```
dl_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['binary_acc
```

In [97]:
```
import datetime

# Generate the logs for Tensorboard
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_fr
```

In [98]:
```
dl_model.fit(
    X_train_pca, y_train,
    validation_data=(X_test_pca, y_test),
    batch_size = 8,
    epochs = 10,
    callbacks=[tensorboard_callback]
)
```

```
Epoch 1/10
5/5 [==============================] - 1s 98ms/step - loss: 5.9499 - binary_accurac
y: 0.6842 - val_loss: 2.3814 - val_binary_accuracy: 0.5882
Epoch 2/10
5/5 [==============================] - 0s 43ms/step - loss: 5.0527 - binary_accurac
y: 0.7105 - val_loss: 1.9399 - val_binary_accuracy: 0.6471
Epoch 3/10
5/5 [==============================] - 0s 41ms/step - loss: 4.1549 - binary_accurac
y: 0.6842 - val_loss: 1.5711 - val_binary_accuracy: 0.6471
Epoch 4/10
5/5 [==============================] - 0s 45ms/step - loss: 3.4173 - binary_accurac
y: 0.6579 - val_loss: 1.2528 - val_binary_accuracy: 0.6471
Epoch 5/10
5/5 [==============================] - 0s 39ms/step - loss: 2.7403 - binary_accurac
y: 0.6579 - val_loss: 1.0283 - val_binary_accuracy: 0.7059
Epoch 6/10
5/5 [==============================] - 0s 39ms/step - loss: 2.0336 - binary_accurac
y: 0.6842 - val_loss: 0.8921 - val_binary_accuracy: 0.7059
Epoch 7/10
5/5 [==============================] - 0s 43ms/step - loss: 1.7104 - binary_accurac
y: 0.7105 - val_loss: 0.7901 - val_binary_accuracy: 0.6765
Epoch 8/10
5/5 [==============================] - 0s 39ms/step - loss: 1.3323 - binary_accurac
y: 0.7895 - val_loss: 0.7309 - val_binary_accuracy: 0.7059
Epoch 9/10
5/5 [==============================] - 0s 39ms/step - loss: 1.1010 - binary_accurac
y: 0.8158 - val_loss: 0.6954 - val_binary_accuracy: 0.7059
Epoch 10/10
5/5 [==============================] - 0s 39ms/step - loss: 0.8917 - binary_accurac
y: 0.8421 - val_loss: 0.6715 - val_binary_accuracy: 0.7059
```

Out[98]:
```
<keras.src.callbacks.History at 0x278a3a52410>
```

In [99]:
```
pred = dl_model.evaluate(X_test_pca, y_test)
pred
```

```
2/2 [==============================] - 0s 0s/step - loss: 0.6715 - binary_accuracy:
0.7059
```

Out[99]:
```
[0.6714609861373901, 0.7058823704719543]
```

In [100…
```
from collections import Counter
from sklearn.metrics import confusion_matrix
import itertools
```

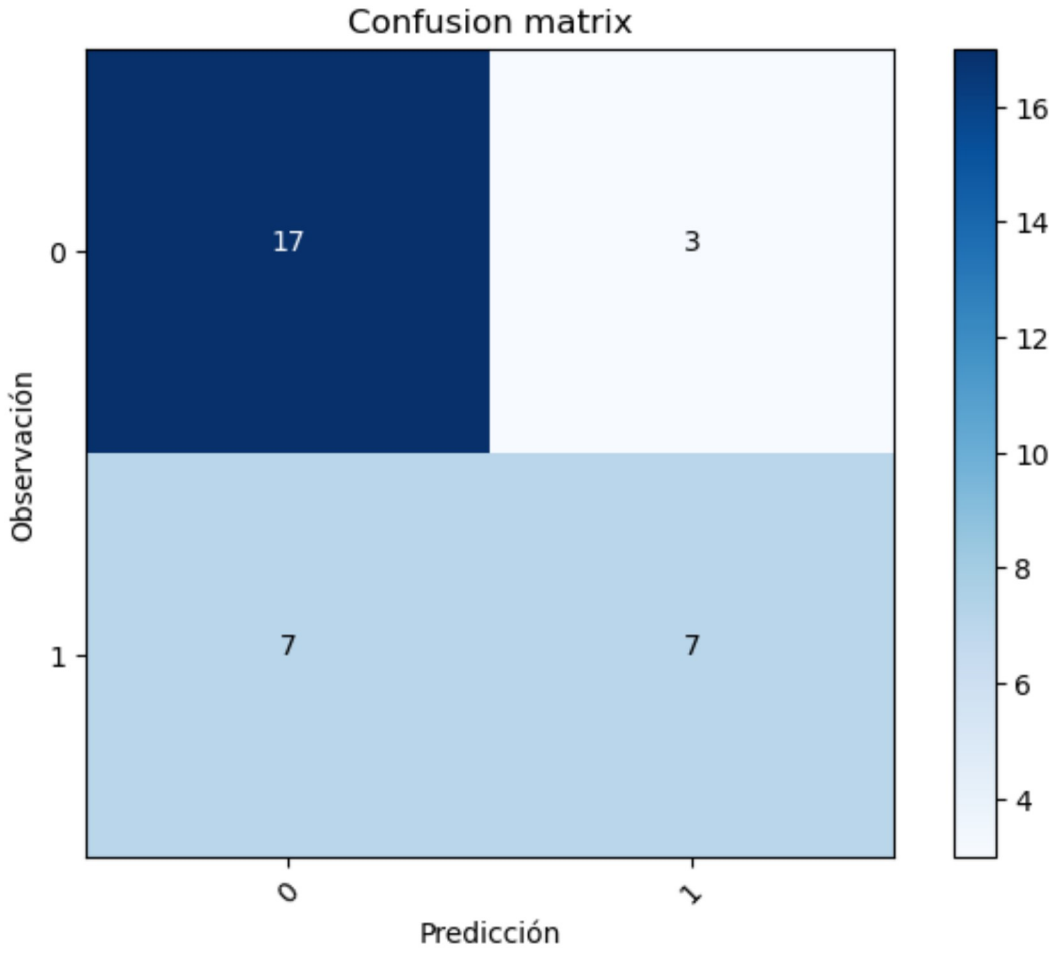
In [101…

```
# Look at confusion matrix
#Note, this code is taken straight from the SKLEARN website, an nice way of viewing
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Observación')
    plt.xlabel('Predicción')
```

In [115…

```
# Predict the values from the validation dataset
y_pred = dl_model.predict(X_test_pca)
# 0.5 Threshold due to sigmoid activation in last layer
y_pred_classes = np.where(y_pred > 0.5, 1, 0)
# compute the confusion matrix
confusion_mtx = confusion_matrix(y_test, y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(2))
```

```
2/2 [==============================] - 0s 0s/step
```



Conclusions

In [108…
```
# It seems like the Neural Network did better in terms of accuracy compared to the
# Though, in both cases, it just seems like it's predominantly picking 0 (ALL) prea
# at random times.
# I am not convinced that these models are any great for actually detecting ALL vs
```

References

- Academic (if any)
- Online (if any)

In [ ]:
```
# None
```

Credits

- If you use and/or adapt your code from existing projects, you must provide links and acknowldge the authors.

> *This code is based on .... (if any)*

In [ ]:
```

```

In [6]:
```
# End of Project
```