We Inspect Team B:
AI Studio Final Presentation

https://yesweinspect.com/
Presentation Date: TBD

Introductions

Meet Our Team!

Bonnie White
CSU Long Beach

Aissatou Thiombane
Arizona State University/UCLA

Nicole Escamilla
UCLA

Andres Vicente
CSULA

Reshma Sheikh
Santa Monica College

Our AI Studio TA and Challenge Advisors

Swagath Babu
AI Studio TA

Corey Levy
Challenge Advisor

Presentation Agenda

1. Problem Statement and Our Goal
2. Business Impact
3. Approach
4. Resources
5. Data Understanding & Data Preparation
6. Modeling & Evaluation
7. Final Thoughts
8. Q&A

AI Studio Project Overview

Construct an ML model able to detect correlations between mold types and associated symptoms, enabling predictive insights into likely health outcomes.

Our Goal

- Our objective is to build an unsupervised learning model that will predict symptoms based on mold types and measurements.

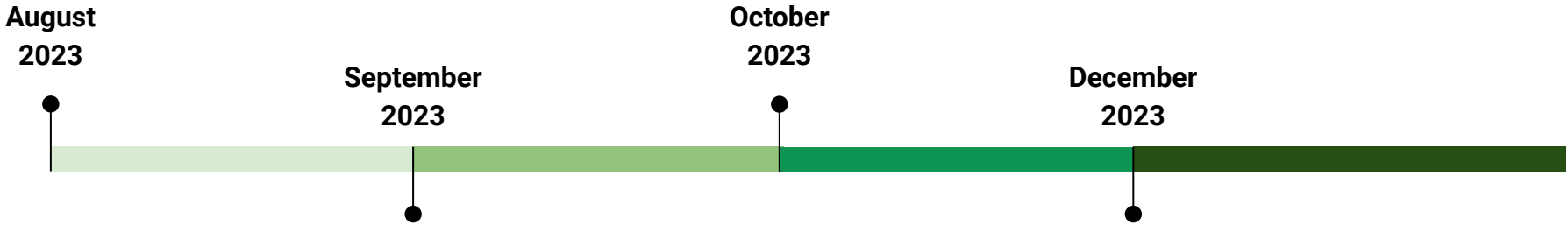
Business Impact

- Assist and accelerate possible mold-induced health diagnoses in the future

- Accessible to the general public & health care providers

- Help address the underlying causes of undiagnosed conditions, rather than merely managing the symptoms

Our Approach



Business Understanding

Spent meetings introducing ourselves and getting to know our teammates, TA, and Challenger Advisor. We took this time to build an understanding of what is expected from each team member, and what the desired end goal looks like.

Data Understanding & Preparation

Exploring data & cleaning spreadsheets to prepare data for visualization, processing, and other preparation methods.

Modeling & Evaluation

Wrapped up data cleaning and preparation to move on to get data model ready.

Final Presentation & Share Findings

Recording our findings, analyzing our results, and observing visualizations. Wrap up project with presentations and sharing our journey.

Resources We Leveraged

- **Data Cleaning**
  - Python
  - Jupyter Notebook
  - Pandas
  - Numpy
- **Data Preparation**
  - Dimension Reduction
  - One-hot encoding
  - PCA
  - Standard Scaler
- **Visualization**
  - Seaborn
  - Matplotlib
- **Modeling**
  - Scikit Learn
    - Random Forest
    - Support Vector Machine
  - Tensorflow
    - FCNN

Data Understanding & Data Preparation

Dataset Overview

- Our dataset can be split into a 4 categories
  - Location Data
  - Symptom Data
  - Health Summary Information
  - Mold Data
- Some columns provided no new information, so it was removed
  - Zip code
    - No need for this since we had City and State data
  - Symptoms
    - All entries had symptoms
- Cleaned all rows by removing extra characters and any variations of N/A
- Ensured all numerical columns had the same int / float data type for processing

One-Hot Encoding

- Our data had string columns with multiple symptoms separated by commas, so we had to create a custom one-hot encoding method

```python
def oneHotEncodeSymptoms(symptomDF):
    # For every column(body system) in symptom DF columns
    for column in symptomDF.columns:
        systemSymptomList = (df[column]
                             .dropna()             # Drop any remaining NaNs (though there shouldn't be any after fillna)
                             .str.lower()          # Convert to lowercase
                             .str.replace(" ", "") # Remove spaces
                             .str.split(',')       # Split by commas
                             .explode()            # Explode lists to rows
                             .unique())            # Get unique values
        # For every symptom in the list, create [body_system]_[symptom]
        # and for every row enter 1 if symptom string exists in element and 0 if not
        for symptom in systemSymptomList:
            newColumn = f"{column}_{symptom}"
            symptomDF[newColumn] = df[column].apply(lambda x: 1 if symptom in str(x).lower().replace(" ", "") else 0)
        systemSymptomList = []
    return symptomDF
```

Dimension Reduction & Visualization

- Our next goal was to explore the cleaned data and see if we would make any initial observations
- Normalized our data with standard scaler method in the Mold and Symptom columns
- After performing one-hot encoding on all categorical columns, we ended up with a data frame consisting of 581 rows and 145 columns, so we need a way to reduce the number of columns to be processed on
- We decided to do some research on PCA to achieve this task
- We applied PCA to both the mold and symptom categories, keeping at least 80% of the variation

```
df.head()
```

	Aspergillus flavus/oryzae	Aspergillus fumigatus	Aspergillus niger	Aspergillus ochraceus	Aspergillus penicillioides	Aspergillus restrictus*	Aspergillus sclerotiorum	Aspergillus sydowii
0	-0.104428	-0.073751	-0.346139	-0.055444	-0.068028	-0.158983	-0.101535	-0.084850
1	-0.104428	-0.073751	-0.348158	-0.055444	-0.067795	-0.198118	-0.101535	-0.084850
2	-0.104428	-0.072315	-0.357242	0.052017	-0.068028	-0.176772	-0.101535	-0.078989
3	-0.104428	-0.073751	-0.360270	-0.055444	-0.068028	-0.198118	-0.101535	-0.084850
4	0.392251	-0.071357	-0.327971	-0.055444	-0.068028	-0.176772	-0.101535	-0.084850

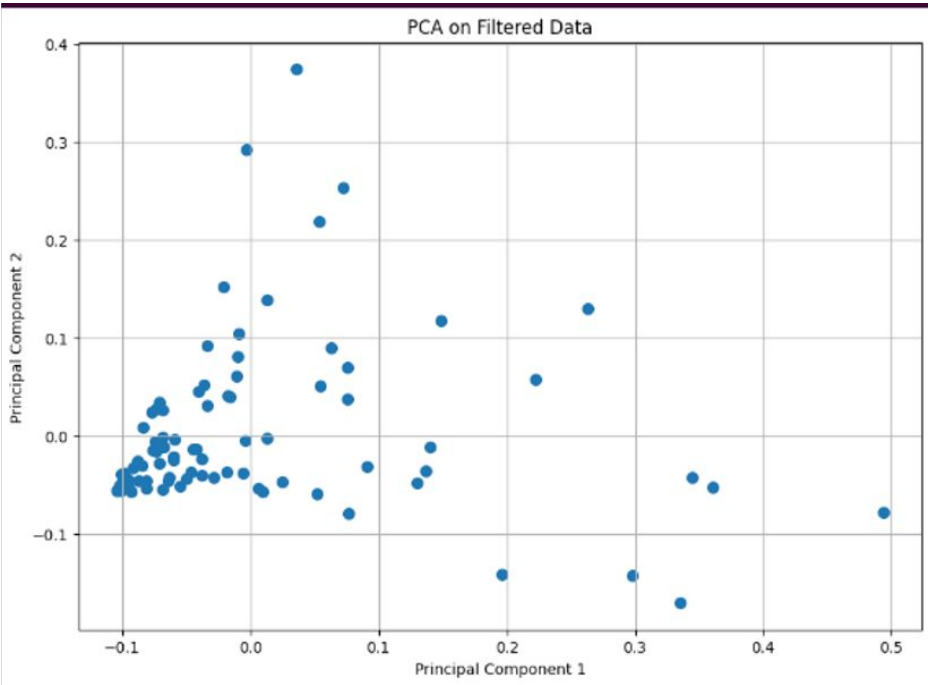
Modeling & Evaluation

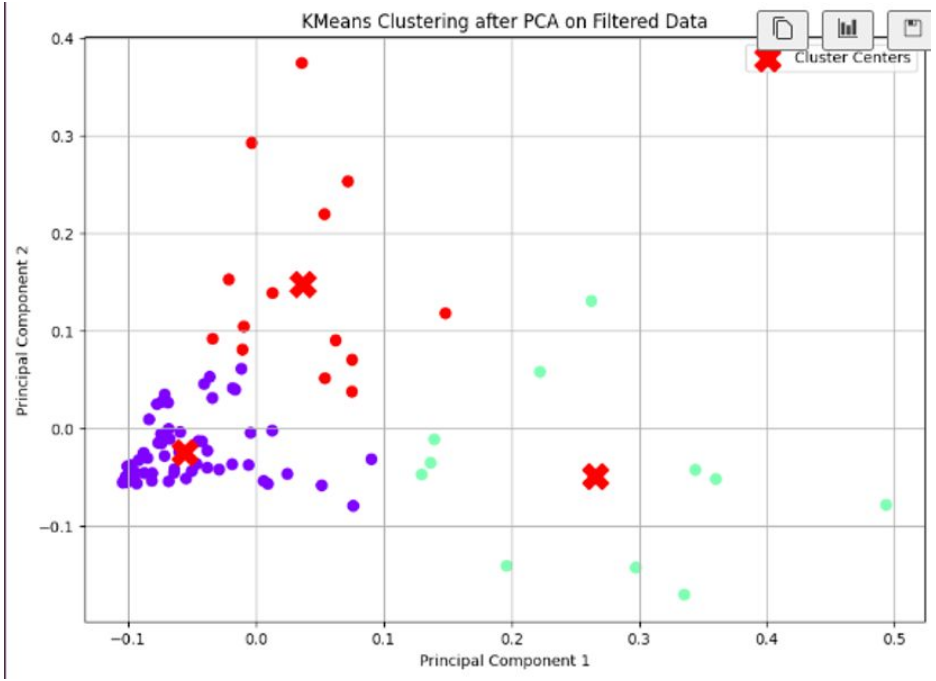
Algorithm Selection

We've begun working on implementing the following algorithms:

- PCA
  - Results shown
- K-Prototypes, K-Means
  - In progress
- Random Forest
  - Results shown
- Support Vector Machine
  - Results shown
- Fully Connected Neural Networks
  - Results shown

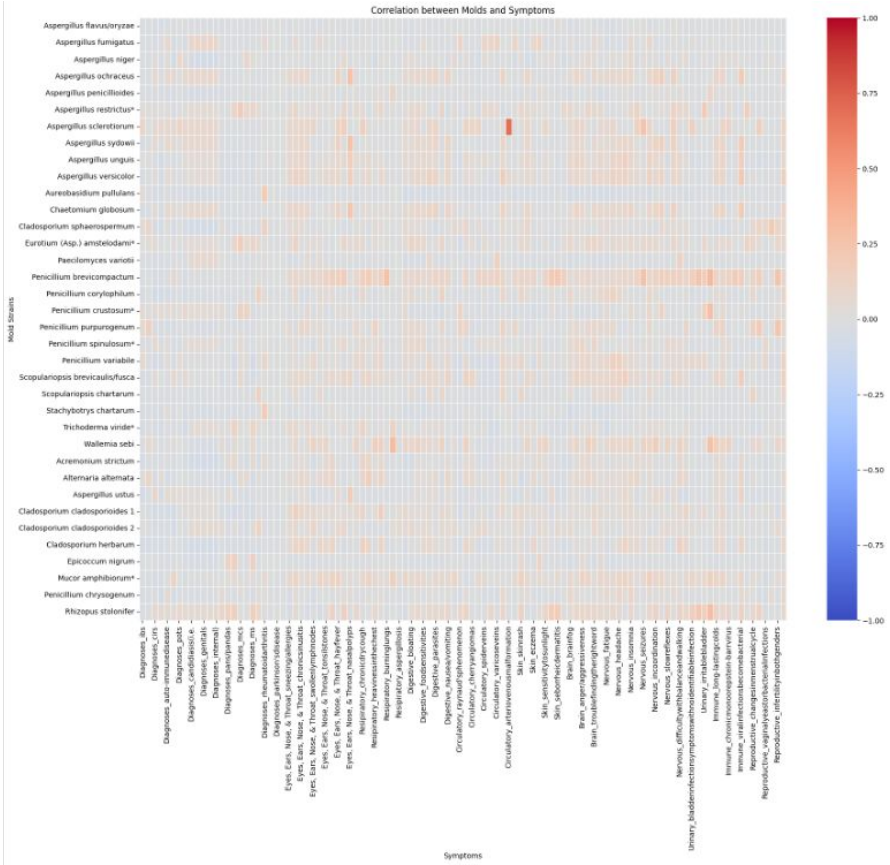
Current Findings (PCA)





After performing one-hot encoding and cleaning our data by separating symptoms, we performed PCA, generating clusters that showed the relationship between different molds and symptoms. This was initially done as a way to reduce dimensionality, visualize relationships, and help with mold to symptom predictions. However, with our data, we have found that other methods like k-prototype may useful as well

Current Findings



Through Data Analysis, we found sparse relationships between our symptoms and molds, hiding our true potential to predict symptoms. Therefore, we readjusted to focusing one individual models to find the true relationship between one body system and the molds. This created multiple models with neater outcomes and coding.

Model Comparison

Model Name	Description	Results	Pros	Cons
Logistic regression	Modeling the probability of a discrete binary outcome	Low accuracy	Easy implementation	Low Accuracy Assumes linearity
Random forest	Modeling the output of multiple decision trees	Higher accuracy than the other models	Addresses discrepancy in data	Time / Slow training
Support Vector machine	Supervised learning algorithm used mostly for classification problems	low accuracy ~ 40% - 55%	Great for addressing discrepancies	Possibility of Underperforming

Model Comparison

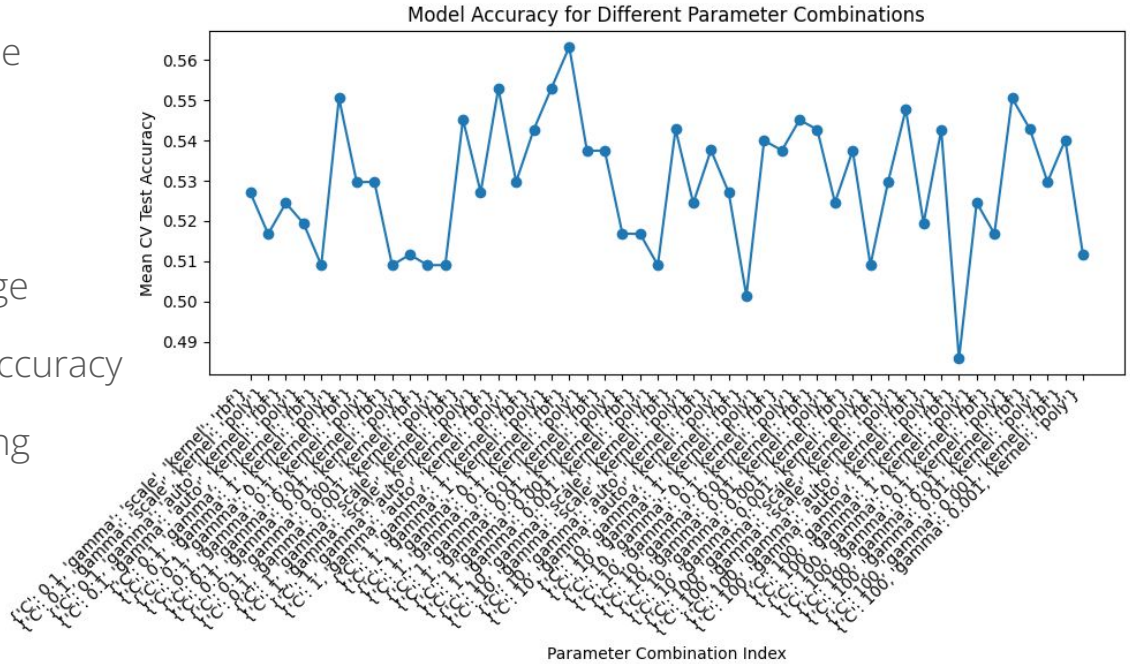
Model Name	Description	Results	Pros	Cons
FCNN	Fully Connected Neural Network.	High Accuracy in some cases ~ 80% - 90% on brain related symptoms ~ 30% - 70% on others or full symptom set	Fast W/ Parallel Processing Scalable Feature Learning	Lots of hyperparameter adjustments needed, not just numerical values Prone to overfitting
GBM	Modeling the output of multiple decision trees	~ 50%	High performance Ensemble learning	Time consuming tuning Overfitting

Model : Support Vector Machine (SVM)

Numerically Categorized Location Data, Numerical Mold values predicting One hot encoded symptoms

Tested on: Brain Symptoms, Nervous System Symptoms, All symptoms

- Accuracy 49% - 56%
- No hyperparameter changes really made too much of a difference
- Changing variance made no difference
- Required PCA
- No linear trend in data
- Increasing prediction labels, no change
- All iterations resulted in under 60% accuracy
- No difference found between including or excluding locational data

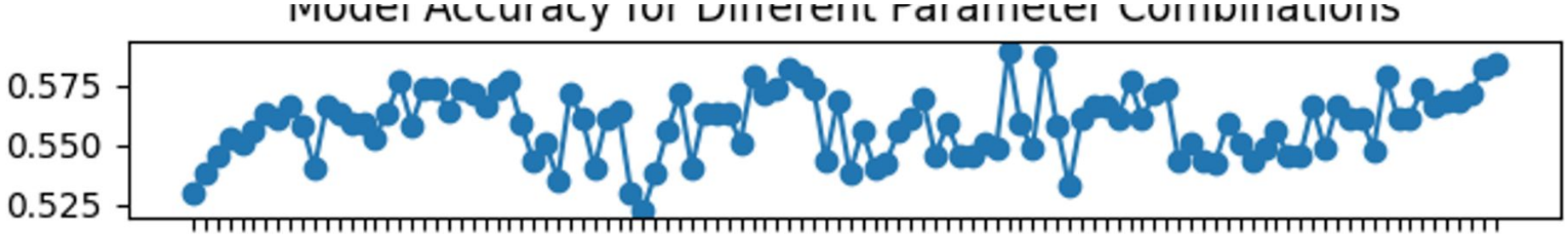


Model : Random Forest

Numerically Categorized Location Data, Numerical Mold values predicting One hot encoded symptoms

Tested on: Brain Symptoms, Nervous System Symptoms, All symptoms

- Very similar results to SVM
- Same patterns recognized
- No hyperparameter tuning responses or trends surfacing
- Location did not have influence, likely because a lack of sample size

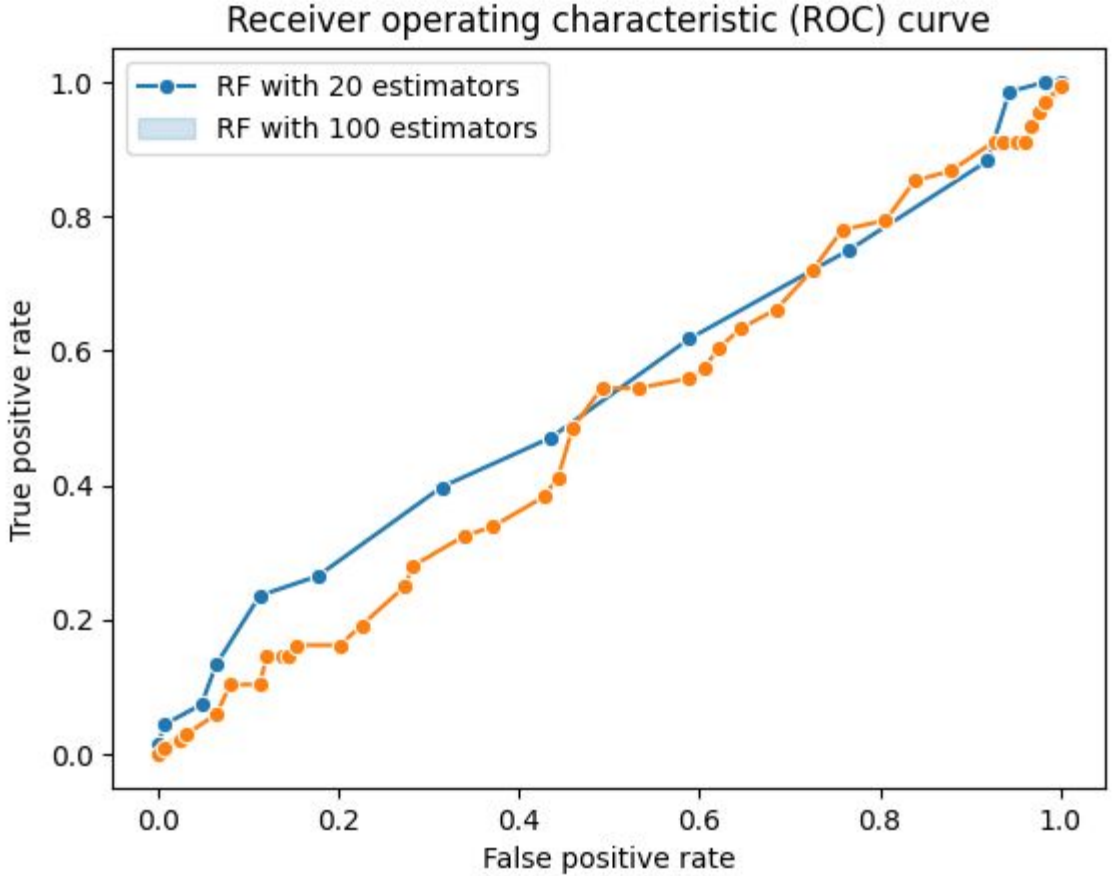


Circulatory Model : Random Forest

```
y_columns = ['Circulatory_spiderveins', 'Circulatory_raynaudsphenomenon','Circulatory_loworreactivebloodpressure','Circulatory_cherryangiomas','Circulatory_easybruising
df['Circulatory'] = df[y_columns].any(axis=1).astype(int)
X_columns = ['Circulatory','Circulatory_spiderveins', 'Circulatory_raynaudsphenomenon','Circulatory_loworreactivebloodpressure','Circulatory_cherryangiomas','Circulator
y= df['Circulatory']
X= df.drop(columns= X_columns, axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1234)
```

```
print('Begin Random Forest Implementation...')
rf_20_model=RandomForestClassifier(criterion='entropy', n_estimators=20)
rf_20_model.fit(X_train, y_train)
rf_20_predictions=rf_20_model.predict_proba(X_test)[:,1].tolist()
rf_100_model=RandomForestClassifier(criterion='entropy', n_estimators=100)
rf_100_model.fit(X_train, y_train)
rf_100_predictions=rf_100_model.predict_proba(X_test)[:,1].tolist()
print('End')

print('Computing ROC Curve...')
fpr_20, tpr_20, thresholds_20=roc_curve(y_test, rf_20_predictions)
fpr_100, tpr_100, thresholds_100=roc_curve(y_test, rf_100_predictions)
print('End')

print('Plotting ROC Curve...')
fig = plt.figure()
ax = fig.add_subplot(111)
sns.lineplot(x=fpr_20, y=tpr_20, marker = 'o')
sns.lineplot(x=fpr_100, y=tpr_100, marker = 'o')
plt.title("Receiver operating characteristic (ROC) curve")
plt.xlabel("False positive rate")
plt.ylabel("True positive rate")
plt.legend(['RF with 20 estimators', 'RF with 100 estimators'])
plt.show()

auc_20=auc(fpr_20, tpr_20)
print("AUC of the RF model with 20 estimators is {:.3f}".format(auc_20))
auc_100=auc(fpr_100, tpr_100)
print("AUC of the RF model with 100 estimators is {:.3f}".format(auc_100))
```

Pytho


Receiver operating characteristic (ROC) curve
RF with 20 estimators
RF with 100 estimators
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

```
    #line6 should predict 1
    ut_data=(0.3922512783666703,-0.0713570261050833,-0.3279711232064132,-0.0554440820674575,-0.06802796
    #row10 should predict 0
    input_data1=(0.1516724533764742,-0.0689630757547511,-0.1987749577362561,-0.0508517359727689,-0.0614488
    #row14 Should Predict 0
    input_data2=(-0.034582120809484,-0.0703994459649504,-0.1634478812405101,-0.0531479090201132,-0.0655496
    #row563 should predict 1
    input_data3=(-0.1044275861292184,-0.0732721863853491,-0.315858982693586,-0.0554440820674575,-0.0680279
    input_data_as_nparray = np.asarray(input_data)
    input_data_as_nparray1 = np.asarray(input_data1)
    input_data_as_nparray2 = np.asarray(input_data2)
    input_data_as_nparray3 = np.asarray(input_data3)
    reshaped_input_data= input_data_as_nparray.reshape(1,-1)
    reshaped_input_data1= input_data_as_nparray1.reshape(1,-1)
    reshaped_input_data2= input_data_as_nparray2.reshape(1,-1)
    reshaped_input_data3= input_data_as_nparray3.reshape(1,-1)
    prediction_of_input_rf = rf_100_model.predict(reshaped_input_data)
    prediction_of_input_rf1 = rf_100_model.predict(reshaped_input_data1)
    prediction_of_input_rf2 = rf_100_model.predict(reshaped_input_data2)
    prediction_of_input_rf3 = rf_100_model.predict(reshaped_input_data3)
    print(prediction_of_input_rf)
    print(prediction_of_input_rf1)
    print(prediction_of_input_rf2)
    print(prediction_of_input_rf3)
```

More...

✓ 0.0s Python

```
[1]
[0]
[0]
[1]
```

Insights and Key Findings

Top Influencial on PC1:
Chaetomium globosum: 0.3856785598848369
Aspergillus versicolor: 0.37945466373278636
Aspergillus sydowii: 0.3671418561467564
Aspergillus ochraceus: 0.361124391806789
Aspergillus unguis: 0.3095465890734753
Scopulariopsis brevicaulis/fusca: 0.2961733809590307
Aspergillus ustus: 0.27200772533117634
Penicillium chrysogenum: 0.17813882082177282
Alternaria alternata: 0.15474839421559902
Acremonium strictum: 0.14628310440546208

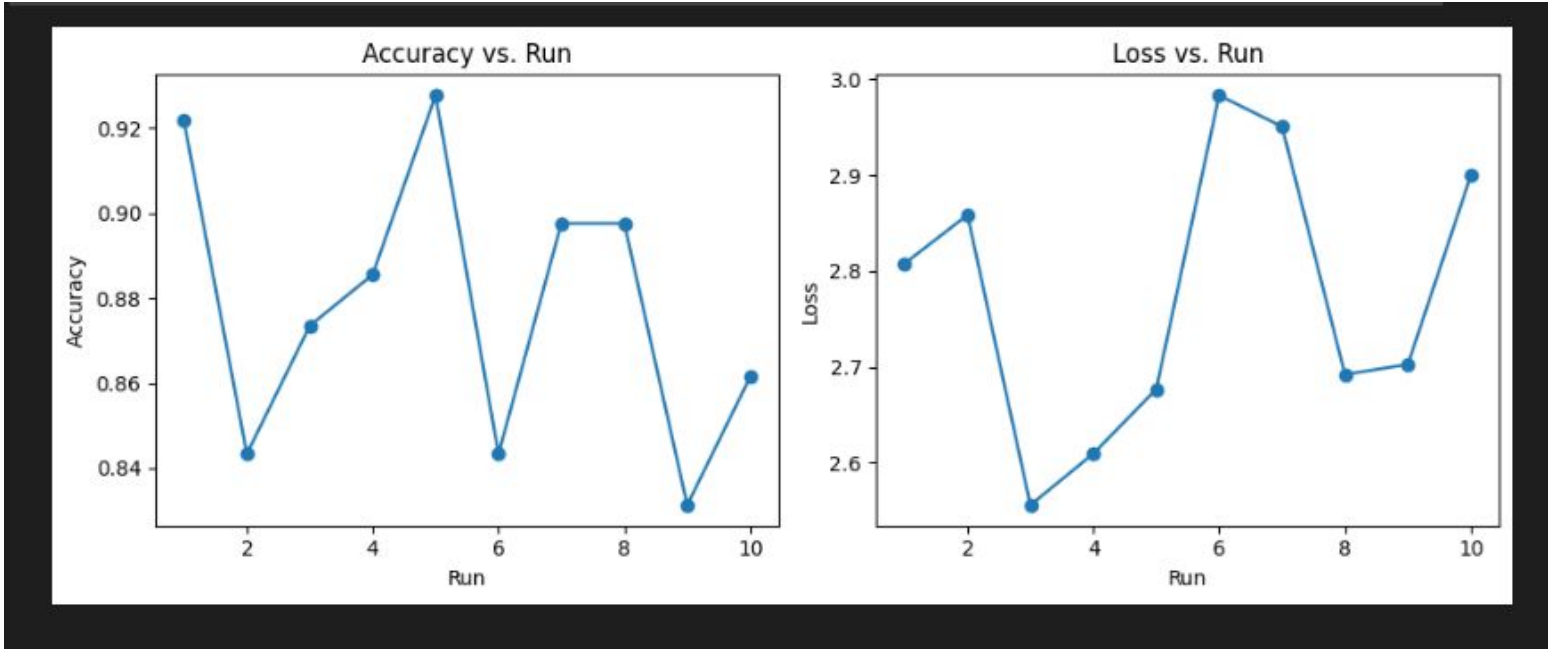
Top Influencial on PC2:
Acremonium strictum: 0.40576436898211093
Epicoccum nigrum: 0.31694155204980495
Aspergillus restrictus*: 0.29819425967327495
Aspergillus ochraceus: -0.25319796892502494
Mucor amphibiorum*: 0.23800159083207065
Aspergillus sydowii: -0.23563630024532337
Cladosporium herbarum: 0.2160421281174504
Aspergillus ustus: -0.20750504001597858
Cladosporium cladosporioides 1: 0.20566992661486438
Alternaria alternata: 0.19950743211227354

Due to the high occurrence of these molds, it is safe to assume that they will be key indicators in Predicting symptoms and diseases.

Model : Fully Connected Neural Networks (FCNNs)

Results from modeling numerical mold values to predict Brain related symptoms:

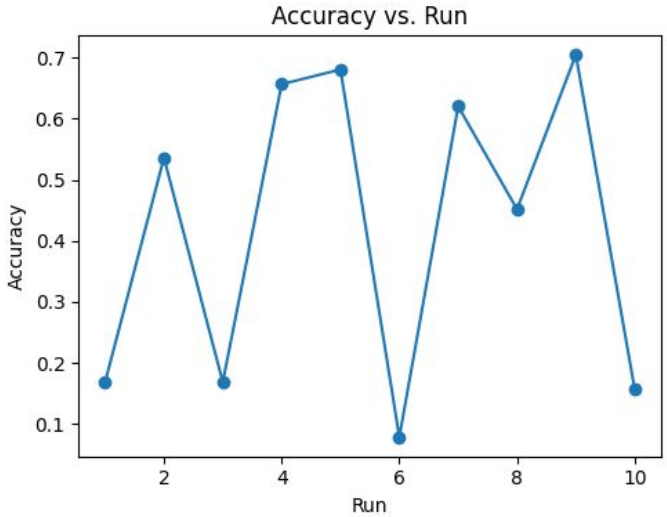
- High Accuracy Range 84% - 92%
- Consistently performing well
- Probably due to the small number of prediction labels (5)

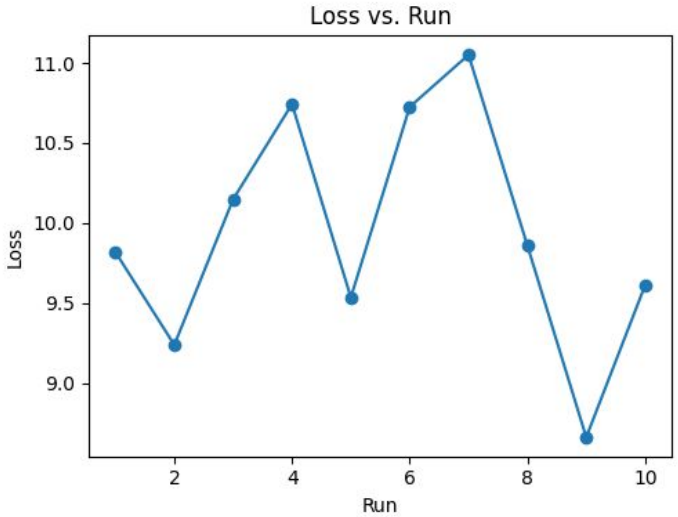


Model : Fully Connected Neural Networks (FCNNs)

Results from modeling numerical mold values to predict Nervous system related symptoms:

- Random Accuracy Range 20% - 70%
- Results vary by and extremely wide range

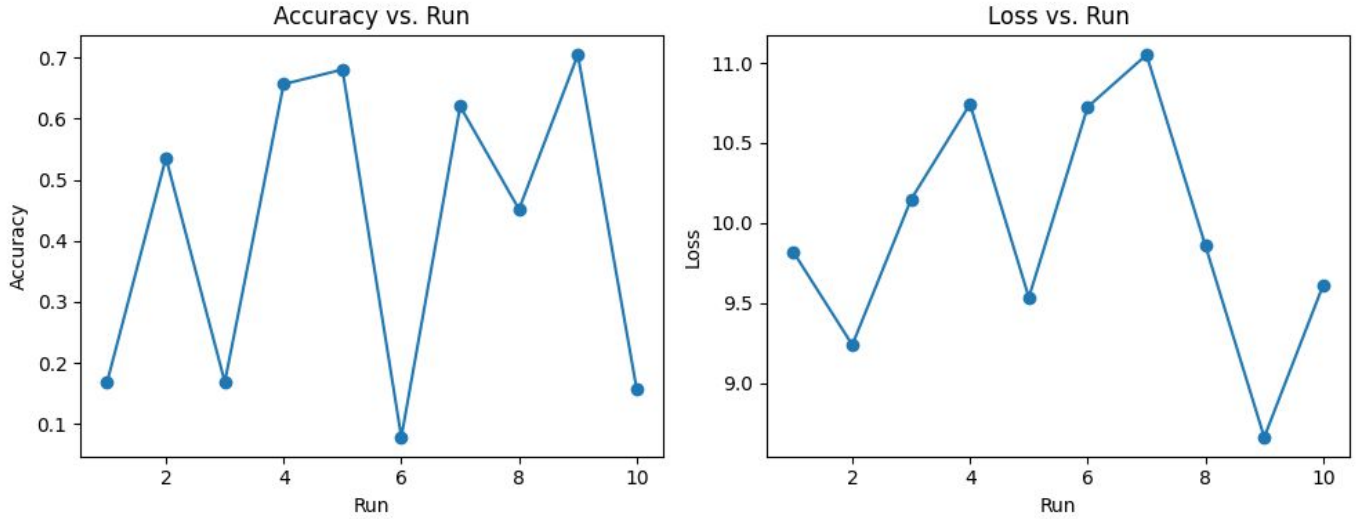




Model : Fully Connected Neural Networks (FCNNs)

Results from modeling numerical mold values to predict all symptoms:

- Random Accuracy Range 20% - 70%
- Results extremely low
- No Accuracy jumping, extremely low
- Trend: The more labels to classify, the harder it is to correctly predict



Model : GBM

Results from modeling numerical mold values to predict all symptoms:

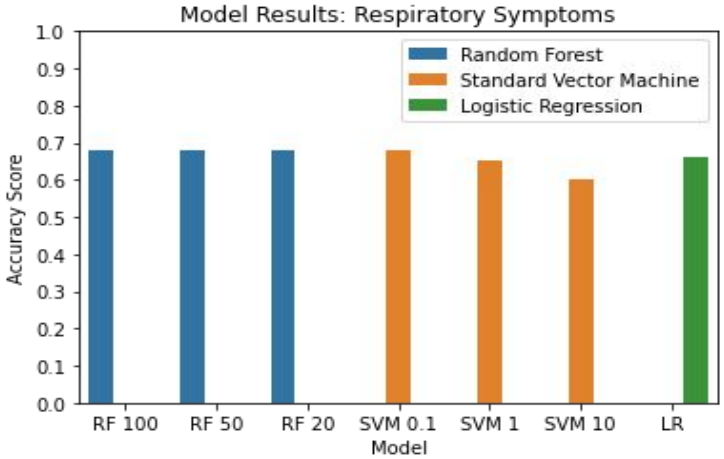
- Accuracy Range 50%
- Seems to be behaving similarly to the other two models

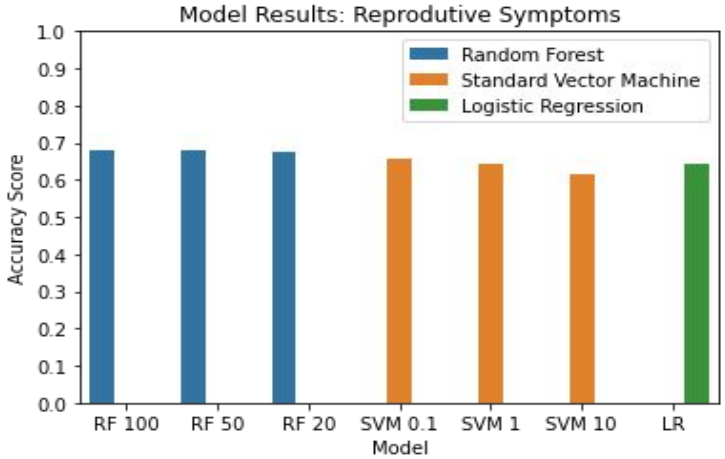
```
Test Accuracy: 0.48795180722891577
              precision    recall  f1-score   support

           0       0.42      0.56      0.48        70
           1       0.58      0.44      0.50        96

    accuracy                           0.49       166
   macro avg       0.50      0.50      0.49       166
weighted avg       0.51      0.49      0.49       166
```

Random Forest, SVM, and Logistic Regression

Analysis on respiratory and reproductive symptoms:

- Labels: ‘Respiratory’ and ‘Reproductive’ symptom columns
  - No one-hot encoding, or location data
- Scores:
  - Random Forest: 0.6807228915662651
  - SVM: 0.6807228915662651
  - Logistic Regression: 0.662650602409638 6





Final Thoughts

What We Learned

- Common Machine Learning tools such as Jupyter Notebooks, Python, Pandas, Numpy, Matplotlib, and Scikit-learn
- The integral steps of building a Machine Learning Model
    - Data Cleaning, Preparation, Visualization, Modeling, and Analysis
- How to approach an unsupervised model
- Handled potential data overload from one-hot encoding using alternative string handling like .split(), .explode(), .unique().
- Attaining Objectives and Delivering Outcomes through Constructive and Collaborative Teamwork
- Modeling process
    - Hyperparameter tuning, outcome record keeping, pattern finding, model architecture research

Potential Next Steps

- Keep training the dataset on the most successful model architecture: FCNNs
- Deploy model
- Build out a tool for public to interact with


Questions?