Shroom Zoom //

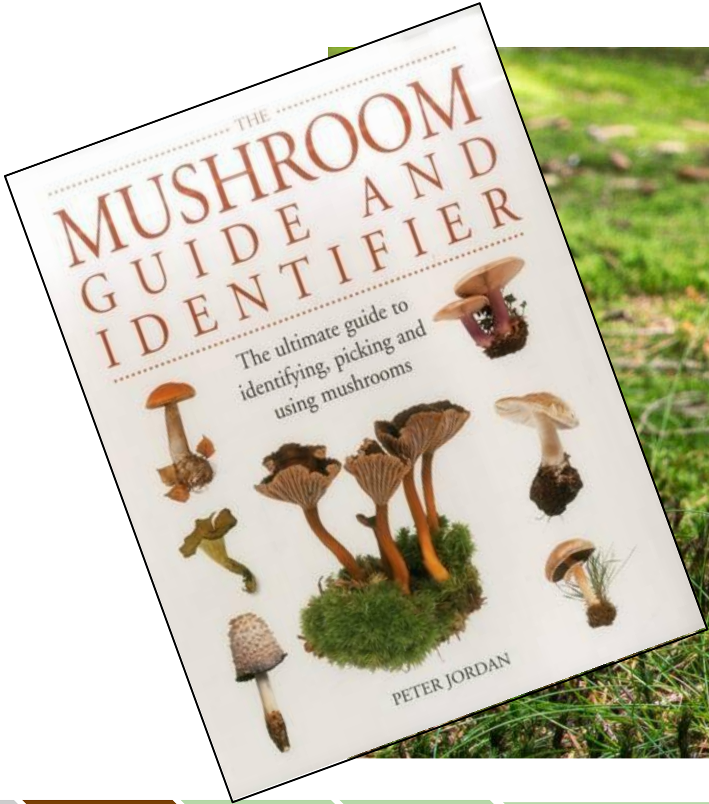
Mushroom Discovery & Edibility Predictor

Team Shrooms:

Denq, Christopher
Manglani, Rajiv
McGrew, Emily
Nguyen, Ben
Ye, Emily

Introduction

Problem



THE MUSHROOM GUIDE AND IDENTIFIER

The ultimate guide to identifying, picking and using mushrooms

PETER JORDAN

Project: Shroom Zoom

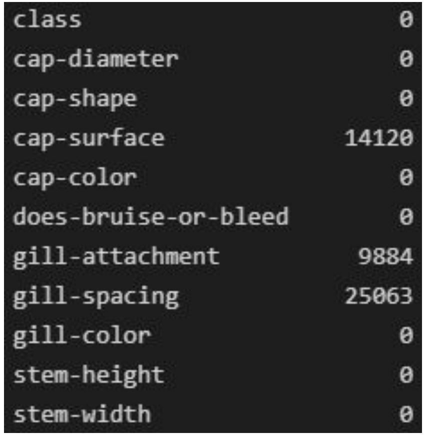
- Full-Stack Web App
  - Predictive Power of Machine Learning
  - 60k data points, 7+ models, 5+ graphs, 3 webpages

- Targeted Google search
  - Identifies mushroom
  - Predicts edibility
  - Presents general fungi facts

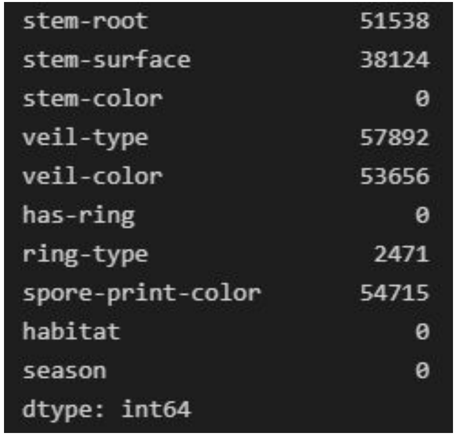
Live Demo

Cleaning

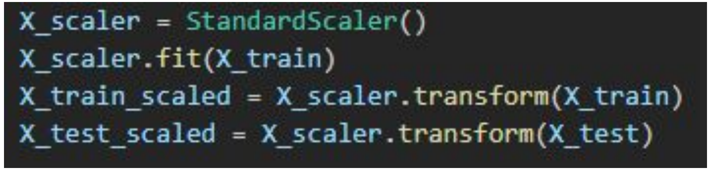
- Dropping Excessive NaN's:
  - ○ Full Columns
  - ○ Rows
- One Hot Encoding
- Split into training/testing data sets and then scale



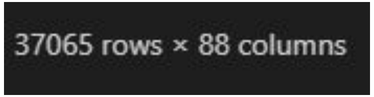
```
class                         0
cap-diameter                  0
cap-shape                     0
cap-surface               14120
cap-color                     0
does-bruise-or-bleed          0
gill-attachment            9884
gill-spacing              25063
gill-color                    0
stem-height                   0
stem-width                    0
```



```
stem-root                 51538
stem-surface              38124
stem-color                    0
veil-type                 57892
veil-color                53656
has-ring                      0
ring-type                  2471
spore-print-color         54715
habitat                       0
season                        0
dtype: int64
```



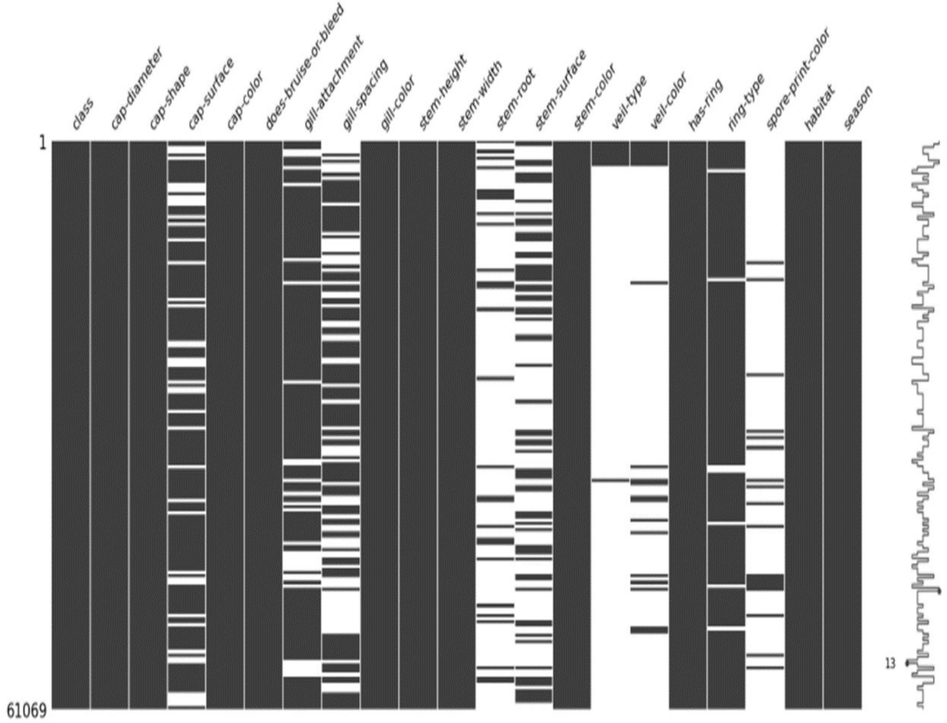
```
X_scaler = StandardScaler()
X_scaler.fit(X_train)
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```



37065 rows × 88 columns

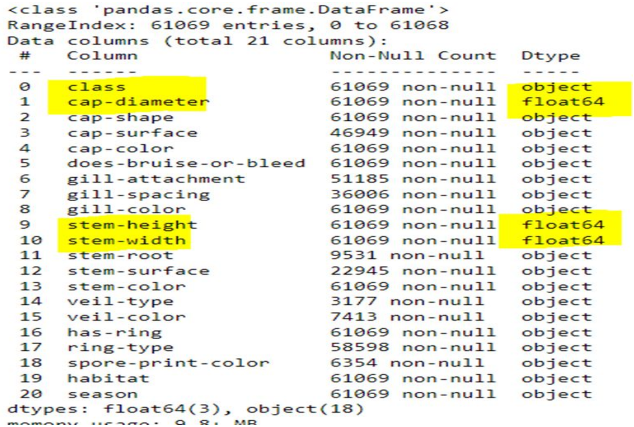
EDA

- Started by importing dependencies such as missingno, pandas, scipy, numpy
- The matrix on the side is reviewing NaN in the data.
- Black columns are string or integer and white is NaN

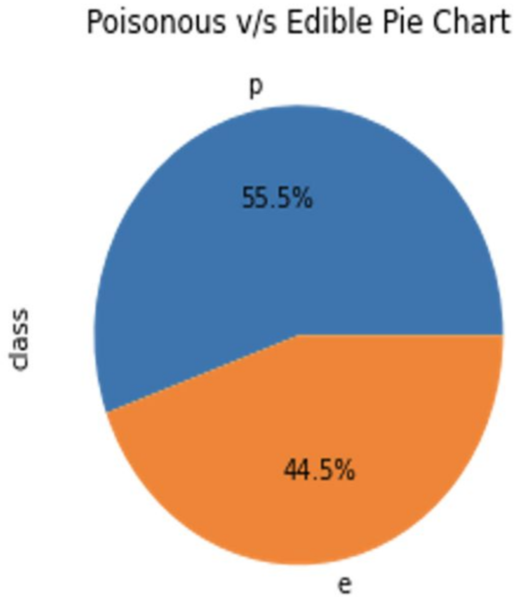


EDA

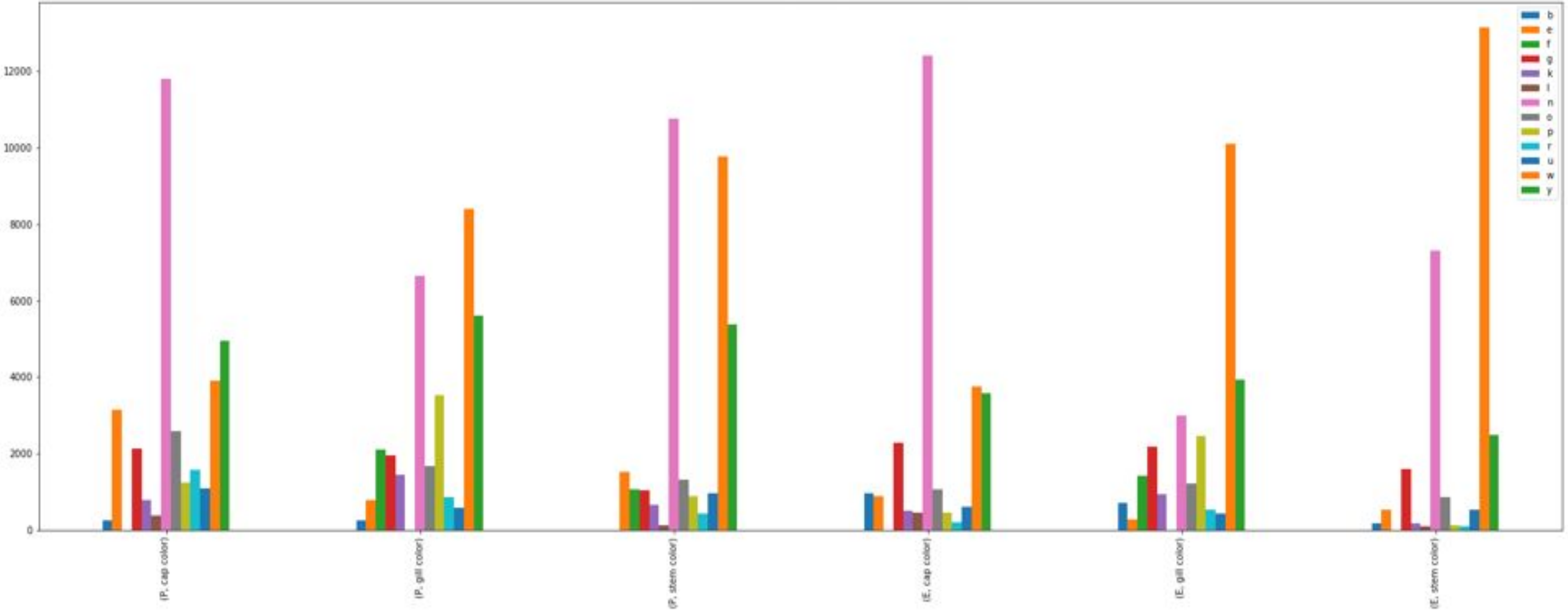
- INFO () FUNCTION TO CHECK FOR FLOAT AND OBJECT



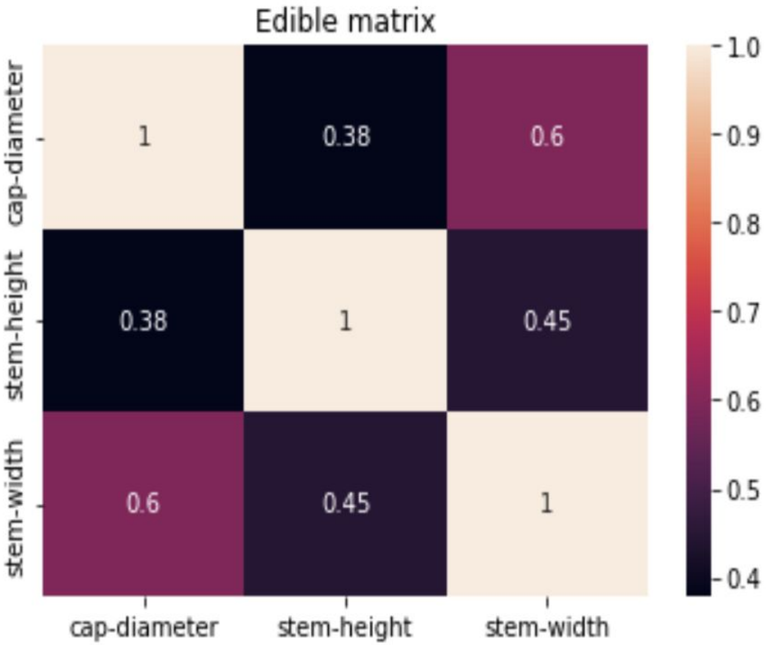
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61069 entries, 0 to 61068
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   class                 61069 non-null  object
 1   cap-diameter          61069 non-null  float64
 2   cap-shape             61069 non-null  object
 3   cap-surface           46949 non-null  object
 4   cap-color             61069 non-null  object
 5   does-bruise-or-bleed  61069 non-null  object
 6   gill-attachment       51185 non-null  object
 7   gill-spacing          36006 non-null  object
 8   gill-color            61069 non-null  object
 9   stem-height           61069 non-null  float64
 10  stem-width            61069 non-null  float64
 11  stem-root             9531 non-null   object
 12  stem-surface          22945 non-null  object
 13  stem-color            61069 non-null  object
 14  veil-type             3177 non-null   object
 15  veil-color            7413 non-null   object
 16  has-ring              61069 non-null  object
 17  ring-type             58598 non-null  object
 18  spore-print-color     6354 non-null   object
 19  habitat               61069 non-null  object
 20  season                61069 non-null  object
dtypes: float64(3), object(18)
memory usage: 9.8+ MB
```

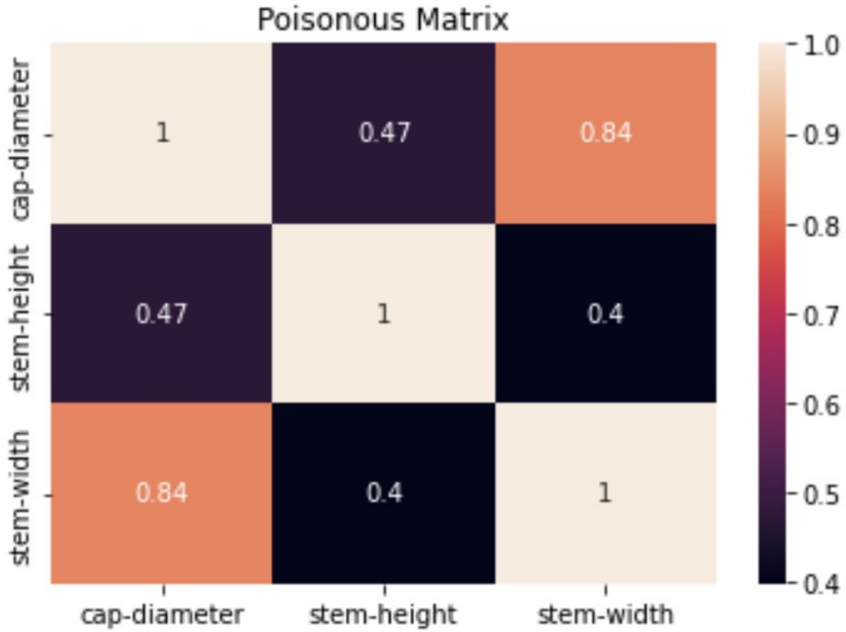


Poisonous vs Edible



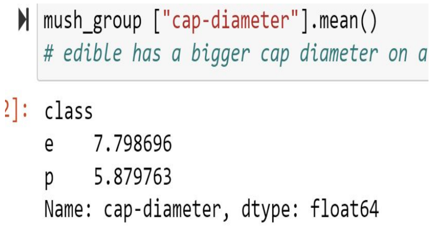
EDA - Matrix


Edible matrix
cap-diameter
stem-height
stem-width
1
0.38
0.6
0.38
1
0.45
0.6
0.45
1
cap-diameter
stem-height
stem-width
1.0
0.9
0.8
0.7
0.6
0.5
0.4


Poisonous Matrix
cap-diameter
stem-height
stem-width
1
0.47
0.84
0.47
1
0.4
0.84
0.4
1
cap-diameter
stem-height
stem-width
1.0
0.9
0.8
0.7
0.6
0.5
0.4

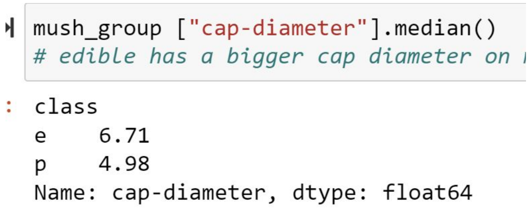
EDA - MEAN & MEDIAN

Edibles have higher Mean and median overall compared to Poisonous



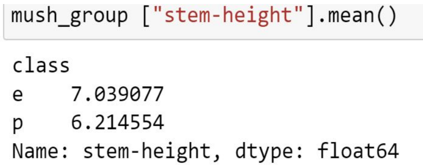
```
mush_group ["cap-diameter"].mean()
# edible has a bigger cap diameter on a
```

```
]: class
   e    7.798696
   p    5.879763
   Name: cap-diameter, dtype: float64
```



```
mush_group ["cap-diameter"].median()
# edible has a bigger cap diameter on
```

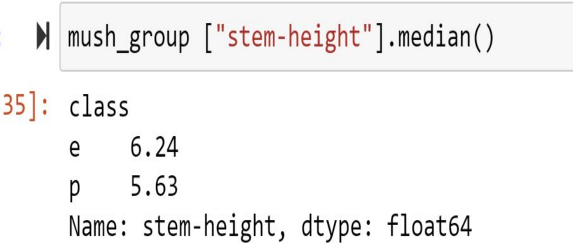
```
: class
  e    6.71
  p    4.98
  Name: cap-diameter, dtype: float64
```



```
mush_group ["stem-height"].mean()
```

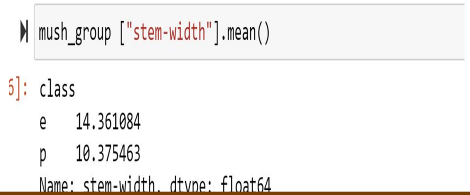
```
class
e    7.039077
p    6.214554
Name: stem-height, dtype: float64
```

```
mush_group ["stem-height"].median()
```



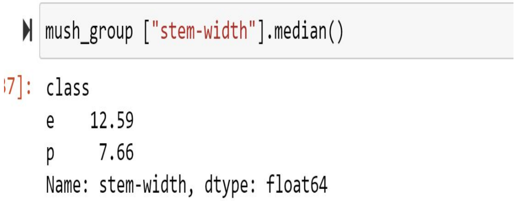
```
mush_group ["stem-height"].median()
```

```
35]: class
     e    6.24
     p    5.63
     Name: stem-height, dtype: float64
```



```
mush_group ["stem-width"].mean()
```

```
5]: class
    e    14.361084
    p    10.375463
    Name: stem-width, dtype: float64
```

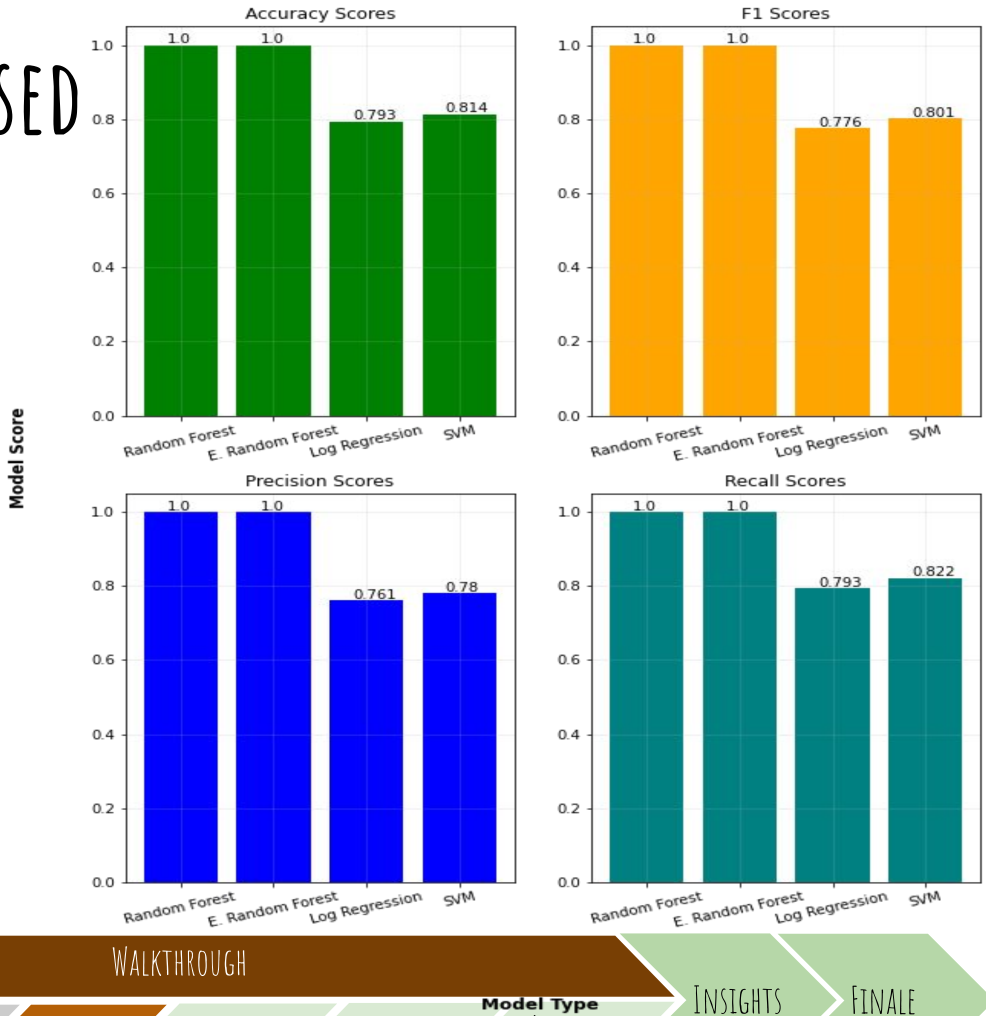


```
mush_group ["stem-width"].median()
```

```
7]: class
    e    12.59
    p    7.66
    Name: stem-width, dtype: float64
```

Machine Learning - Supervised

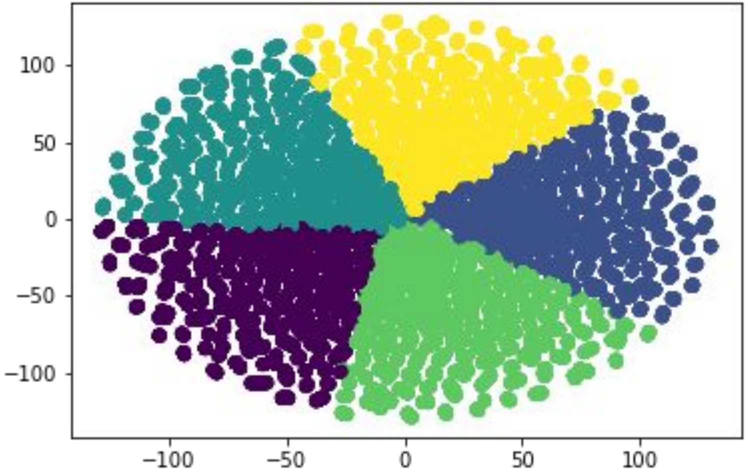
- Logistic regression
- SVM
- KNN
- Random forest
- Extreme random forest



Machine Learning - UnSupervised

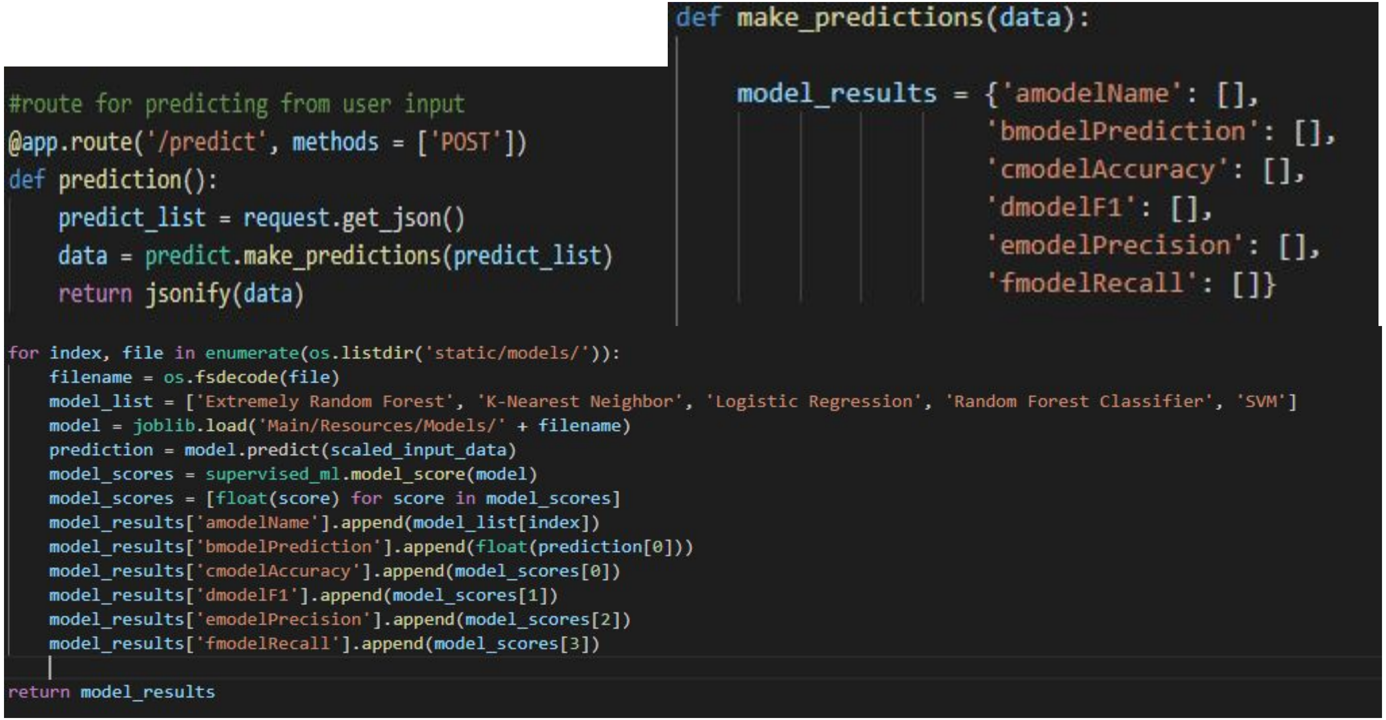
- Future Recommendation sys
  - PCA, TSNE
  - DBSCAN
  - KMEANS - more evenly sized clusters

KMeans clusters (n=5)
(post PCA + TSNE)



Server Connection and Deployment

- Flask Server
- Predict Function
  - ○ transform Input data
  - ○ Load models
  - ○ Build Dictionary



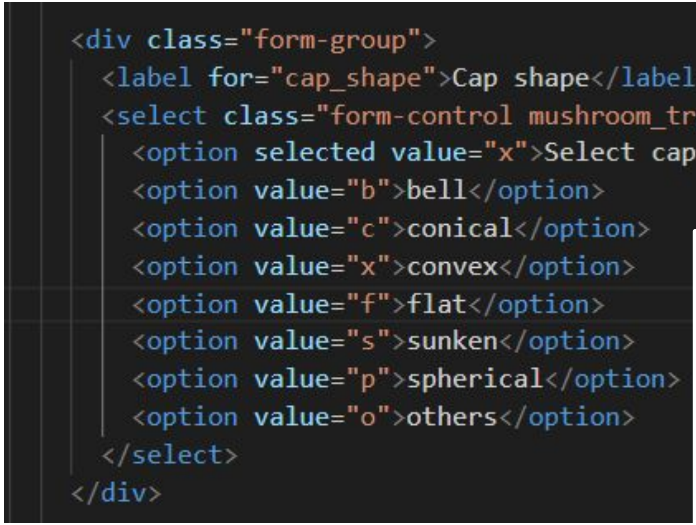
```
#route for predicting from user input
@app.route('/predict', methods = ['POST'])
def prediction():
    predict_list = request.get_json()
    data = predict.make_predictions(predict_list)
    return jsonify(data)
```

```
def make_predictions(data):

    model_results = {'amodelName': [],
                     'bmodelPrediction': [],
                     'cmodelAccuracy': [],
                     'dmodelF1': [],
                     'emodelPrecision': [],
                     'fmodelRecall': []}
```

```
for index, file in enumerate(os.listdir('static/models/')):
    filename = os.fsdecode(file)
    model_list = ['Extremely Random Forest', 'K-Nearest Neighbor', 'Logistic Regression', 'Random Forest Classifier', 'SVM']
    model = joblib.load('Main/Resources/Models/' + filename)
    prediction = model.predict(scaled_input_data)
    model_scores = supervised_ml.model_score(model)
    model_scores = [float(score) for score in model_scores]
    model_results['amodelName'].append(model_list[index])
    model_results['bmodelPrediction'].append(float(prediction[0]))
    model_results['cmodelAccuracy'].append(model_scores[0])
    model_results['dmodelF1'].append(model_scores[1])
    model_results['emodelPrecision'].append(model_scores[2])
    model_results['fmodelRecall'].append(model_scores[3])

return model_results
```

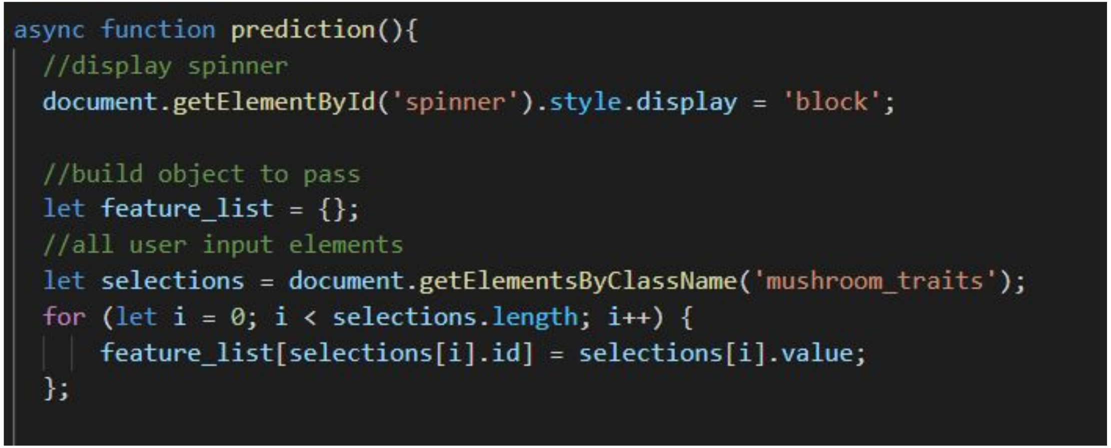
Web Application: Design

HTML



```
<div class="form-group">
  <label for="cap_shape">Cap shape</label>
  <select class="form-control mushroom_traits" id="cap-shape">
    <option selected value="x">Select cap shape</option>
    <option value="b">bell</option>
    <option value="c">conical</option>
    <option value="x">convex</option>
    <option value="f">flat</option>
    <option value="s">sunken</option>
    <option value="p">spherical</option>
    <option value="o">others</option>
  </select>
</div>
```

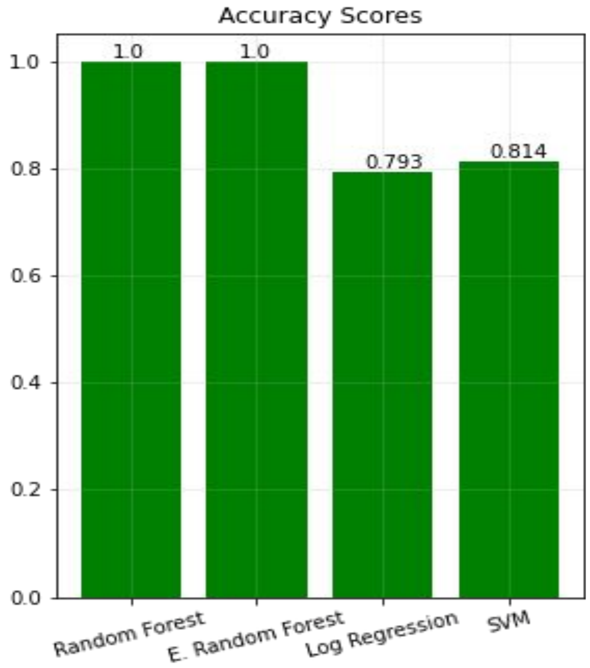
JavaScript



```
async function prediction(){
  //display spinner
  document.getElementById('spinner').style.display = 'block';

  //build object to pass
  let feature_list = {};
  //all user input elements
  let selections = document.getElementsByClassName('mushroom_traits');
  for (let i = 0; i < selections.length; i++) {
      feature_list[selections[i].id] = selections[i].value;
  };
```

Elephant in the room: 1.0?

- Data source
  - 10k real mushrooms -> synthetic 60k undiscovered mushrooms
  - Nature of data is categorical + a large number of NANs
  - RFC and ERFC is extremely applicable here



Next Steps

- Computer vision for photo integration
- Expansion to all flora
- Recommendation system for similar (discovered) mushrooms
- CSV export for scientists
- Feature reduction for easier interpretation

Thank you!