CS M148 – Introduction to Data Science

Project 3 Report

Yaman Yucel

605704529

Introduction:

The main goal of this project is to investigate the National Archives on Mushrooms and build a training machine learning model to distinguish whether a mushroom is edible or poisonous. If we can find that a mushroom is edible, we will use that mushroom in our daily mushroom soup.

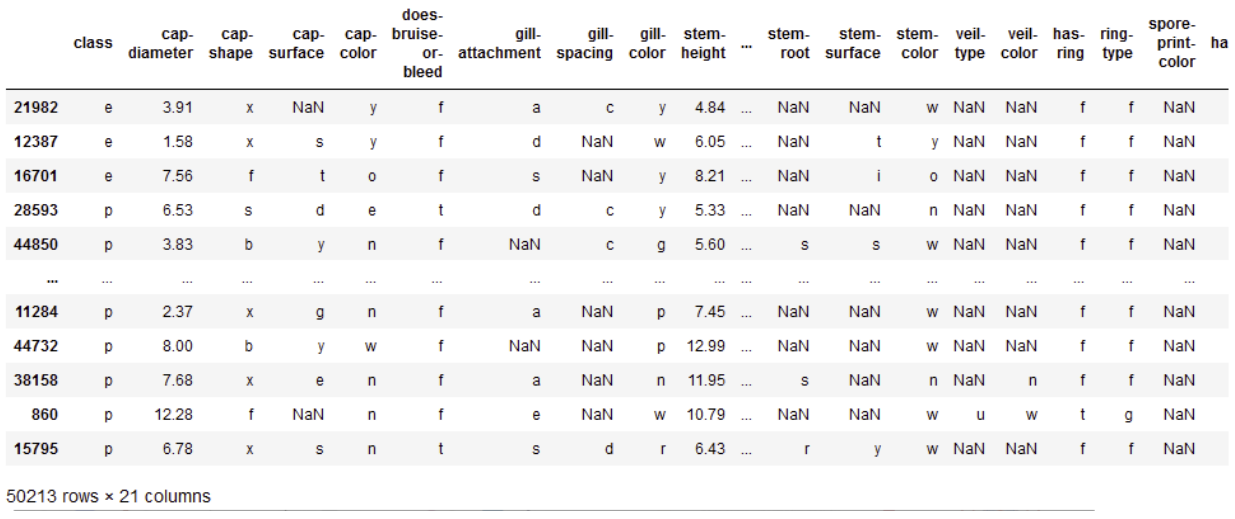
The project consists of 4 parts which are:

- Introduction
- Methodology
  - Data Loading, Splitting, Exploration and Visualization
  - Data Pre-Processing
  - Data Augmentation
  - Statistical Hypothesis Testing
  - Models of your choice
  - Ensemble Method
  - Hyper-parameter Tuning
- Results
- Conclusion

Methodology:

1) Data Loading, Splitting, Exploration and Visualization:

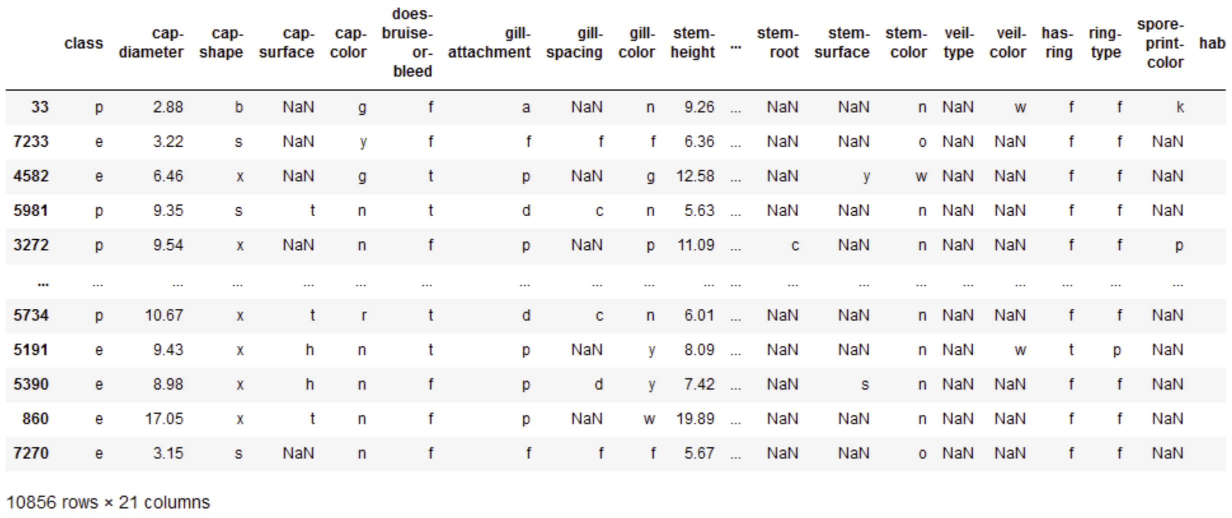
Firstly, the training dataset and testing dataset is investigated. Before, investigation I have shuffled each dataset in the random state 42.



	class	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-spacing	gill-color	stem-height	...	stem-root	stem-surface	stem-color	veil-type	veil-color	has-ring	ring-type	spore-print-color	ha
21982	e	3.91	x	NaN	y	f	a	c	y	4.84	...	NaN	NaN	w	NaN	NaN	f	f	NaN	
12387	e	1.58	x	s	y	f	d	NaN	w	6.05	...	NaN	t	y	NaN	NaN	f	f	NaN	
16701	e	7.56	f	t	o	f	s	NaN	y	8.21	...	NaN	i	o	NaN	NaN	f	f	NaN	
28593	p	6.53	s	d	e	t	d	c	y	5.33	...	NaN	NaN	n	NaN	NaN	f	f	NaN	
44850	p	3.83	b	y	n	f	NaN	c	g	5.60	...	s	s	w	NaN	NaN	f	f	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11284	p	2.37	x	g	n	f	a	NaN	p	7.45	...	NaN	NaN	w	NaN	NaN	f	f	NaN	
44732	p	8.00	b	y	w	f	NaN	NaN	p	12.99	...	NaN	NaN	w	NaN	NaN	f	f	NaN	
38158	p	7.68	x	e	n	f	a	NaN	n	11.95	...	s	NaN	n	NaN	n	f	f	NaN	
860	p	12.28	f	NaN	n	f	e	NaN	w	10.79	...	NaN	NaN	w	u	w	t	g	NaN	
15795	p	6.78	x	s	n	t	s	d	r	6.43	...	r	y	w	NaN	NaN	f	f	NaN	

50213 rows × 21 columns

Figure 1: Training dataset



	class	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-spacing	gill-color	stem-height	...	stem-root	stem-surface	stem-color	veil-type	veil-color	has-ring	ring-type	spore-print-color	hab
33	p	2.88	b	NaN	g	f	a	NaN	n	9.26	...	NaN	NaN	n	NaN	w	f	f	k	
7233	e	3.22	s	NaN	y	f	f	f	f	6.36	...	NaN	NaN	o	NaN	NaN	f	f	NaN	
4582	e	6.46	x	NaN	g	t	p	NaN	g	12.58	...	NaN	y	w	NaN	NaN	f	f	NaN	
5981	p	9.35	s	t	n	t	d	c	n	5.63	...	NaN	NaN	n	NaN	NaN	f	f	NaN	
3272	p	9.54	x	NaN	n	f	p	NaN	p	11.09	...	c	NaN	n	NaN	NaN	f	f	p	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5734	p	10.67	x	t	r	t	d	c	n	6.01	...	NaN	NaN	n	NaN	NaN	f	f	NaN	
5191	e	9.43	x	h	n	t	p	NaN	y	8.09	...	NaN	NaN	n	NaN	w	t	p	NaN	
5390	e	8.98	x	h	n	f	p	d	y	7.42	...	NaN	s	n	NaN	NaN	f	f	NaN	
860	e	17.05	x	t	n	f	p	NaN	w	19.89	...	NaN	NaN	n	NaN	NaN	f	f	NaN	
7270	e	3.15	s	NaN	n	f	f	f	f	5.67	...	NaN	NaN	o	NaN	NaN	f	f	NaN	

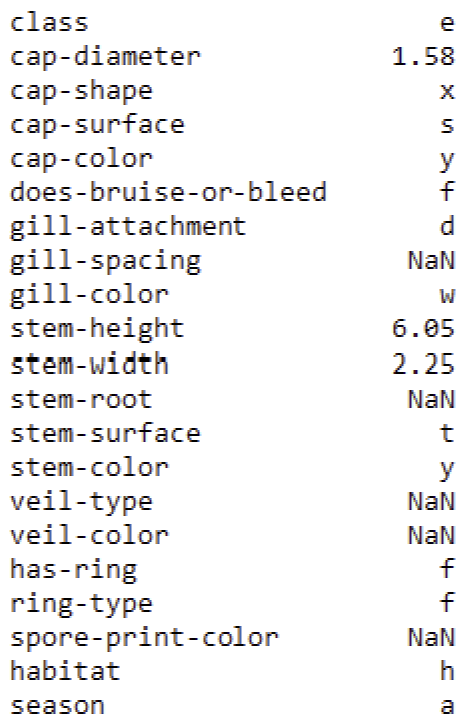
10856 rows × 21 columns

Figure 2: Testing dataset

Remarks at first glance:

1) Some samples contain null values for some specific columns.
2) The dataset is mixture of categorical variables and numerical variables
3) There are 20 features in datasets
4) There are 50213 samples for training dataset and 10856 samples for testing dataset
5) Label class is string and needs transformation to binary values
6) Categorical variables are in the string format and need transformation.

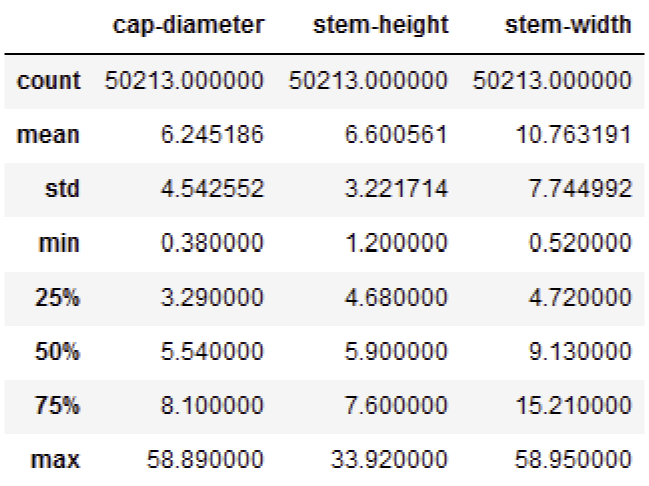
- Then, the training and testing dataset should be splitted into labels and data. However, I will use the splitted data after Data Augmentation.
- Single sample is observed:



```
class                          e
cap-diameter                1.58
cap-shape                      x
cap-surface                    s
cap-color                      y
does-bruise-or-bleed           f
gill-attachment                d
gill-spacing                 NaN
gill-color                     w
stem-height                 6.05
stem-width                  2.25
stem-root                    NaN
stem-surface                   t
stem-color                     y
veil-type                    NaN
veil-color                   NaN
has-ring                       f
ring-type                      f
spore-print-color            NaN
habitat                        h
season                         a
```

Figure 3: Sample 12387 in training dataset

- For numerical features, the describe() method is used to investigate distribution of numerical data.



	cap-diameter	stem-height	stem-width
count	50213.000000	50213.000000	50213.000000
mean	6.245186	6.600561	10.763191
std	4.542552	3.221714	7.744992
min	0.380000	1.200000	0.520000
25%	3.290000	4.680000	4.720000
50%	5.540000	5.900000	9.130000
75%	8.100000	7.600000	15.210000
max	58.890000	33.920000	58.950000

Figure 4: Statistics of Numerical Features

- Each feature is plotted with a histogram if the feature is numerical or plotted with a bar plot if the feature is categorical.

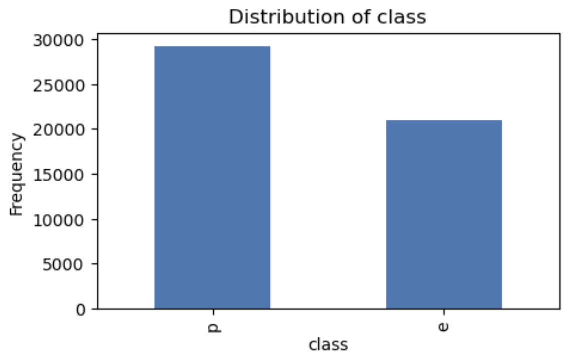


Figure 5: Distribution of labels

- Note that, classes are nearly balanced, therefore class imbalance won't be a significant issue in analysis.

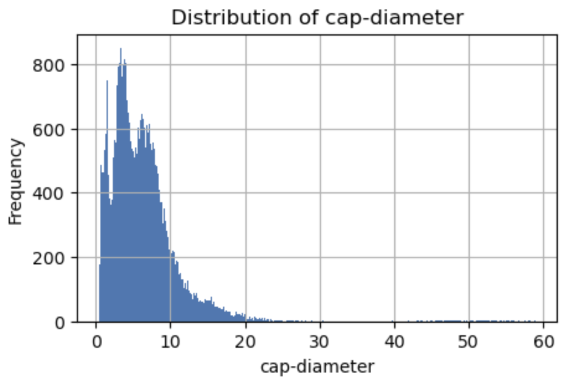


Figure 6: Distribution of cap-diameter, cap-diameter follows a gaussian distribution

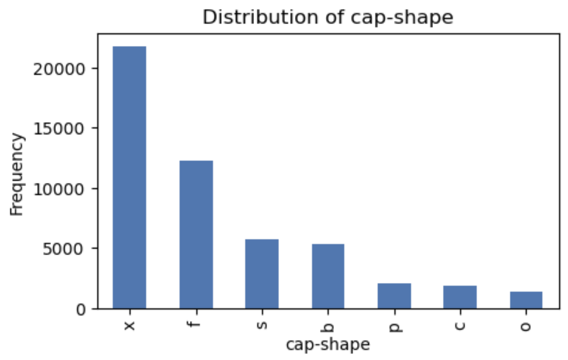


Figure 7: Distribution of cap-shape, feature is unbalanced which can create problem in model performance

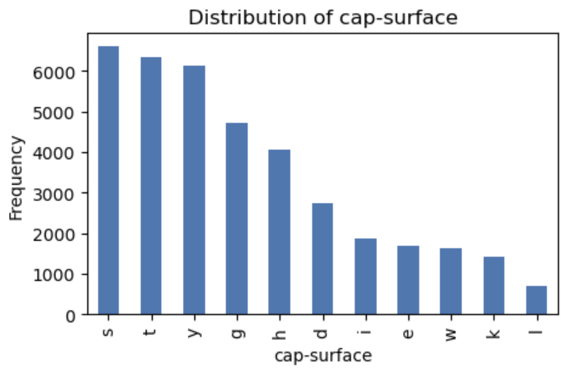


Figure 8: Distribution of cap-surface, feature is unbalanced which can create problem in model performance

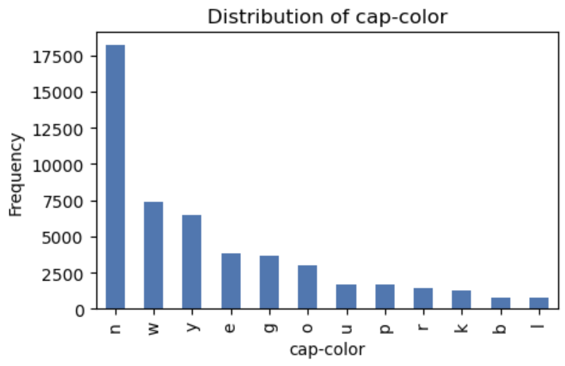


Figure 8: Distribution of cap-color, feature is unbalanced which can create problem in model performance

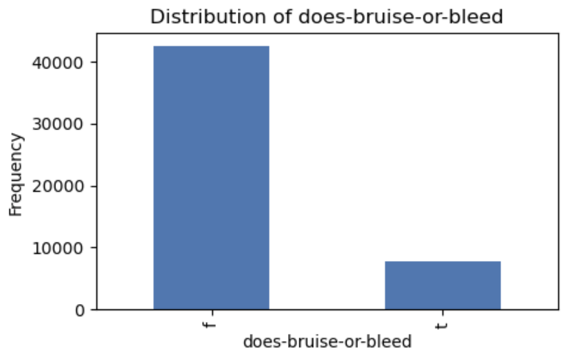


Figure 9: Distribution of does-bruise-or-bleed, feature is unbalanced which can create problem in model performance

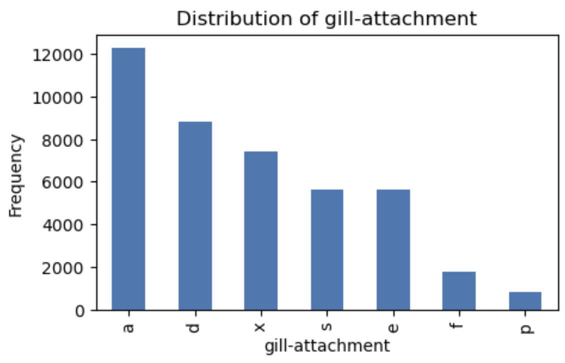


Figure 10: Distribution of gill-attachment, feature is unbalanced which can create problem in model performance

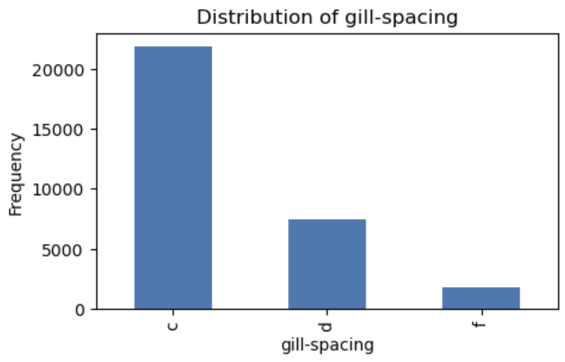


Figure 11: Distribution of gill-spacing, feature is unbalanced which can create problem in model performance

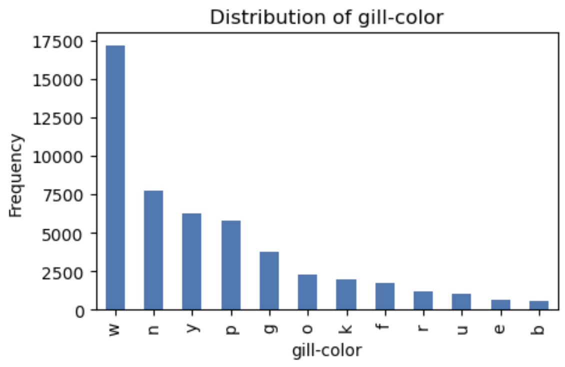


Figure 12: Distribution of gill-color, feature is unbalanced which can create problem in model performance

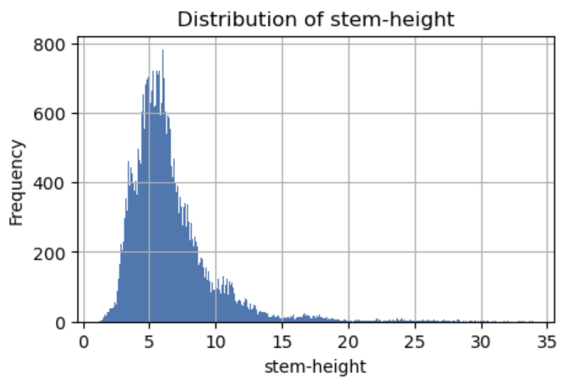


Figure 13: Distribution of stem-height, data follows a gaussian distribution

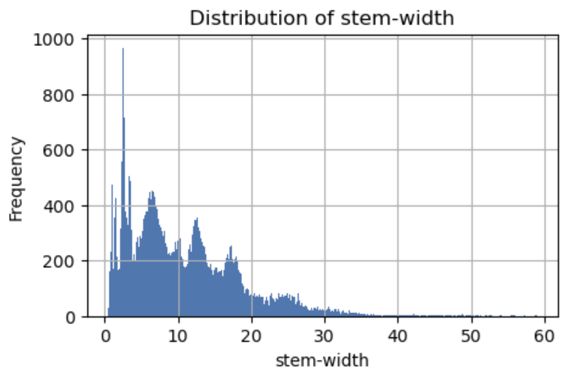


Figure 14: Distribution of stem-width, data slightly follows a gaussian distribution

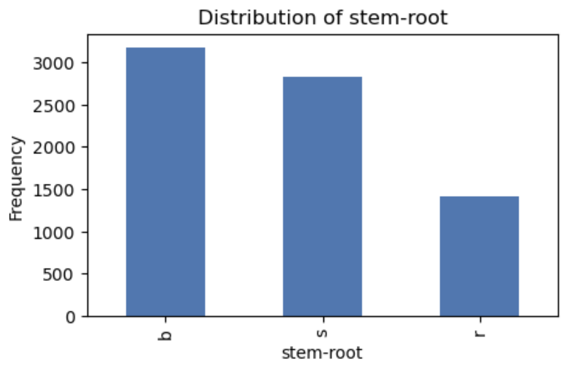


Figure 15: Distribution of stem-root, this feature is nearly balanced which will be helpful.

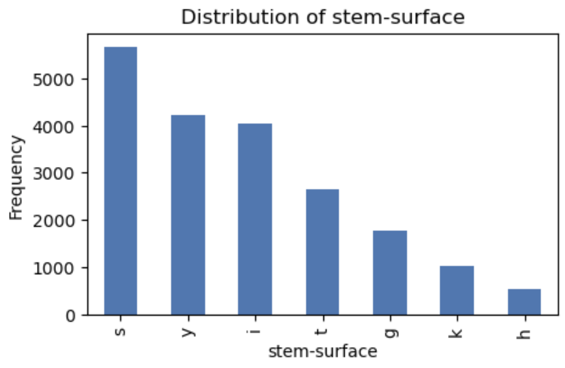


Figure 16: Distribution of stem-surface, feature is unbalanced which can create problem in model performance

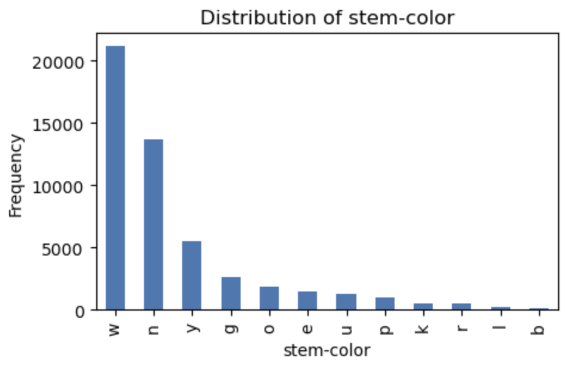


Figure 17: Distribution of stem-color, feature is unbalanced which can create problem in model performance

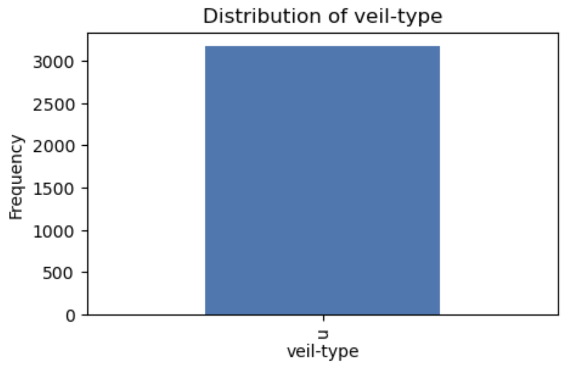


Figure 18: Distribution of veil-type, we will see that veil-type lacks too many data points. There is no universal mushroom in our dataset.

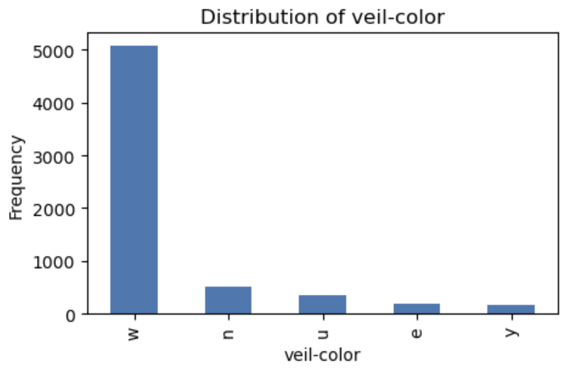


Figure 19: Distribution of veil-color, feature is unbalanced which can create problem in model performance

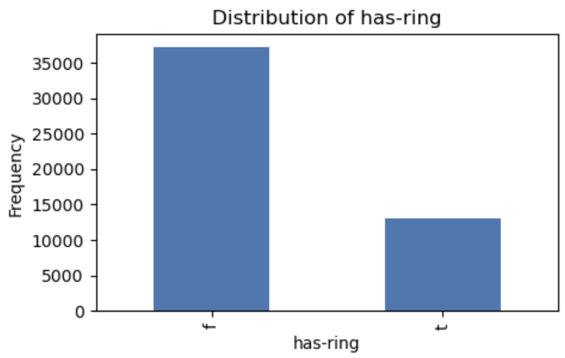


Figure 20: Distribution of has-ring, feature is unbalanced which can create problem in model performance

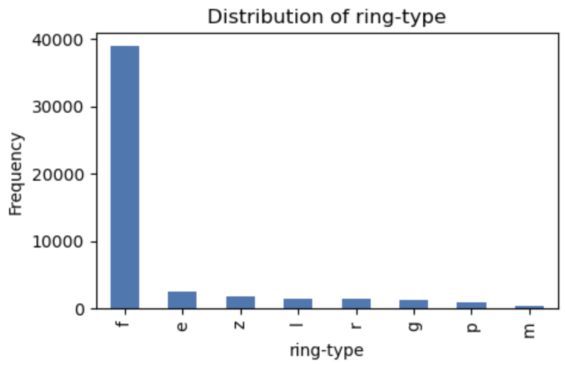


Figure 21: Distribution of ring-type, feature is unbalanced which can create problem in model performance

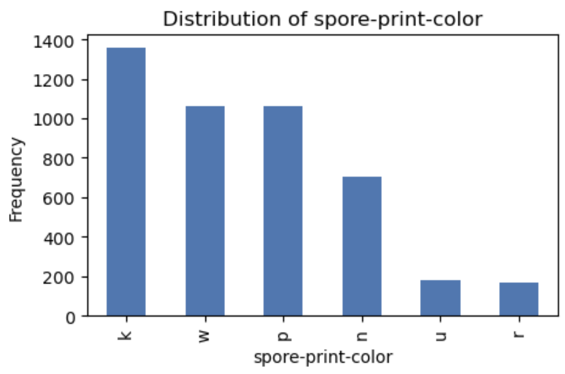


Figure 22: Distribution of spore-print-color, feature is unbalanced which can create problem in model performance

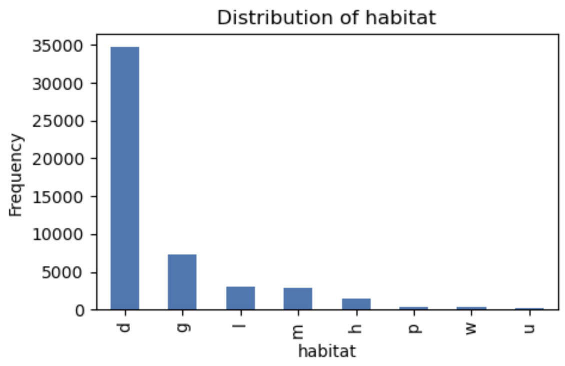


Figure 23: Distribution of habitat, feature is unbalanced which can create problem in model performance

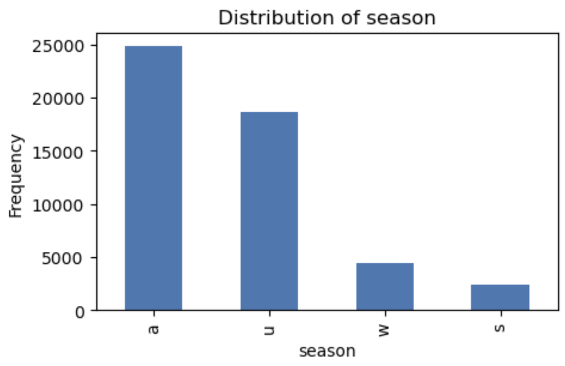


Figure 24: Distribution of season, feature is unbalanced which can create problem in model performance

- Bar and histogram plots without any distinction of classes show that categorical features consist of imbalance in features. Numerical features follows a gaussian distribution.
- In the following part, I have plotted categorical features with distinction of classes to see which feature is useful for classification.

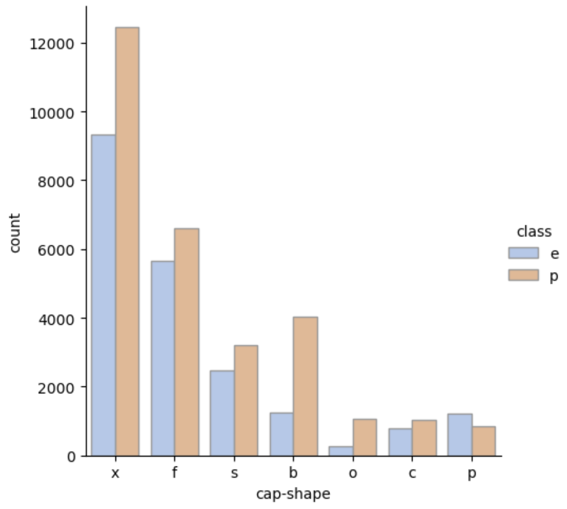


Figure 25: Distribution of cap-shape, if cap-shape is o or b, it is highly likely that the mushroom is poisonous.

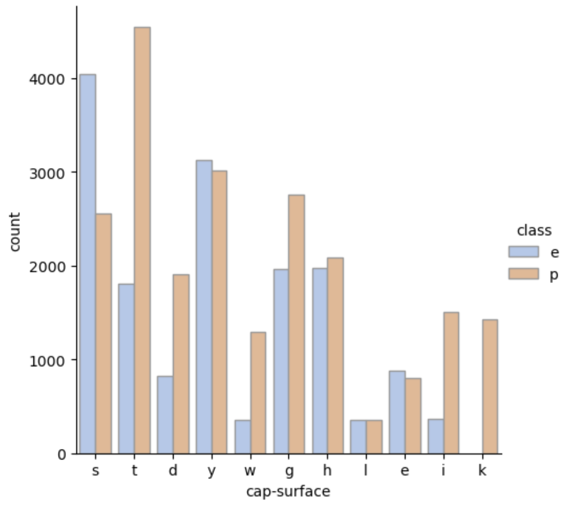


Figure 26: Distribution of cap-surface, if cap-surface is t, d, w, i or k, it is likely that the mushroom is poisonous. If cap-surface is s, it is likely that the mushroom is edible.

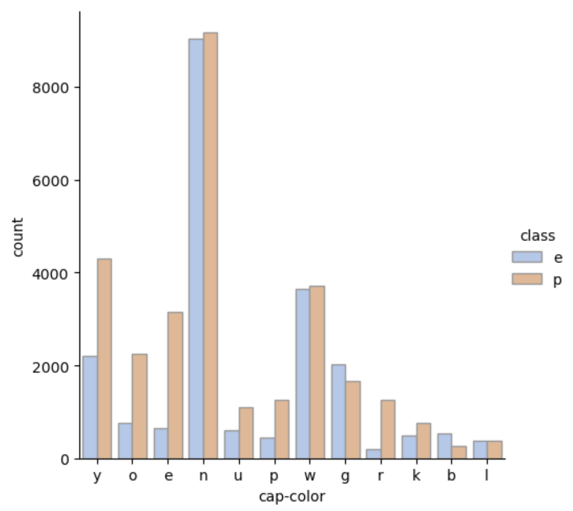


Figure 27: Distribution of cap-color, if cap-color is y, o, e, i, u, p, r, it is likely that the mushroom is poisonous.

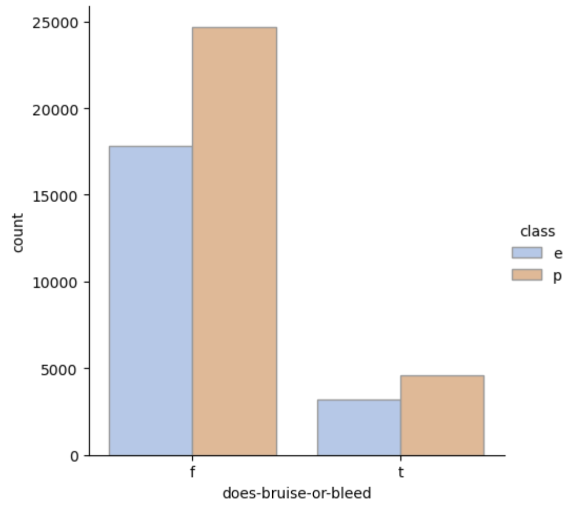


Figure 28: Distribution of does-bruise-or-bleed. There is no distinctive difference.

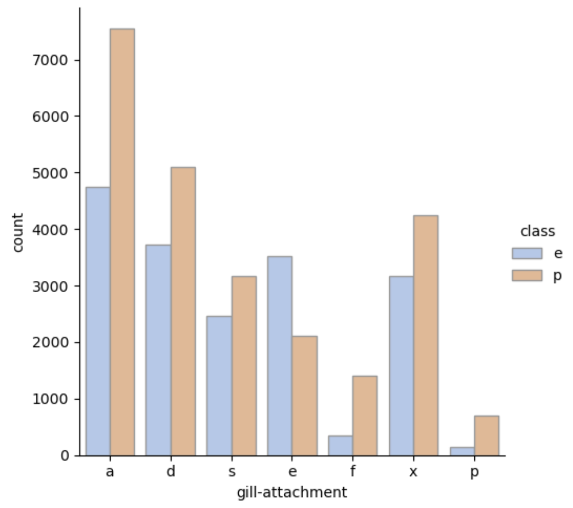


Figure 29: Distribution of gill-attachment. If gill-attachment is a, f or p, it is likely that mushroom is p.

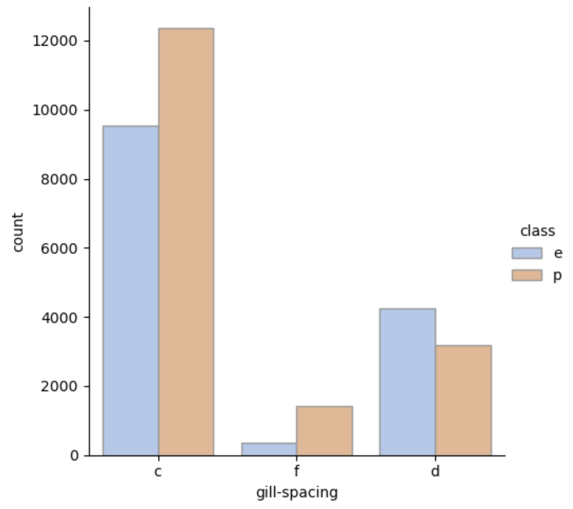


Figure 30: Distribution of gill-spacing. If gill-spacing is f, it is likely that mushroom is p.

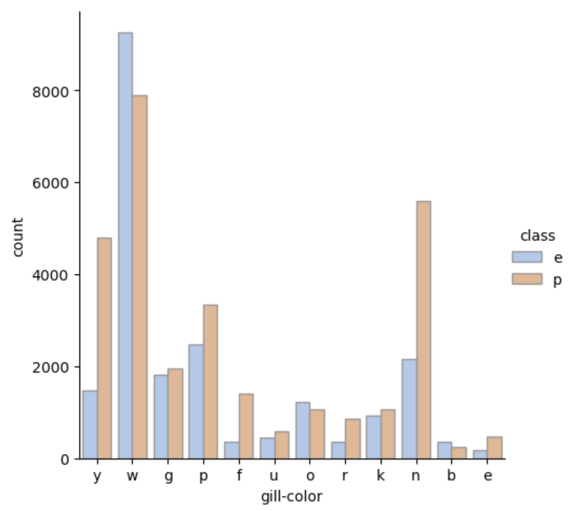


Figure 31: Distribution of gill-color. If gill-color is y, f, r, n, it is likely that mushroom is p.

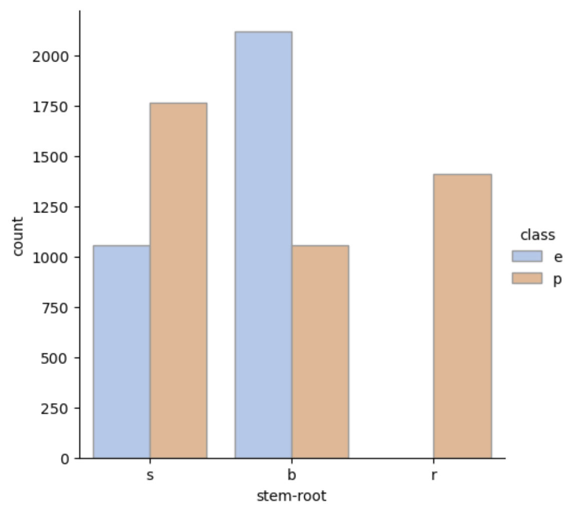


Figure 32: Distribution of stem-root. If stem-root is r it is highly likely that mushroom is p.

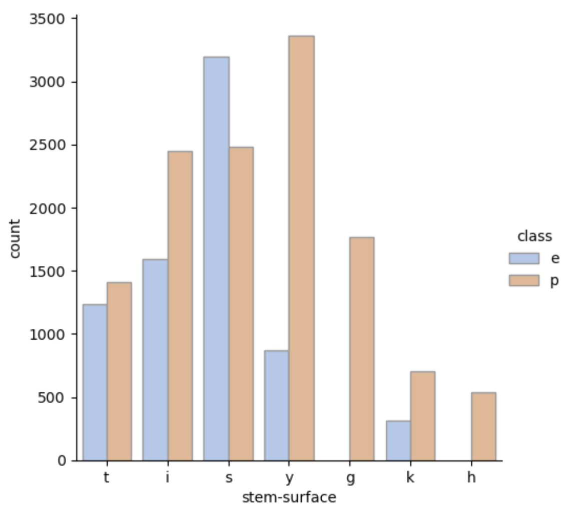


Figure 33: Distribution of stem-surface. If stem-surface is y,g or h it is highly likely that mushroom is p.

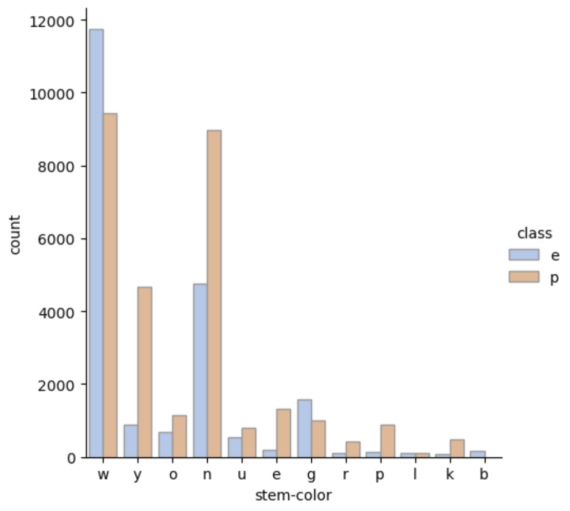


Figure 32: Distribution of stem-color. If stem-color is y, n, u, e, p or k, it is highly likely that mushroom is p.

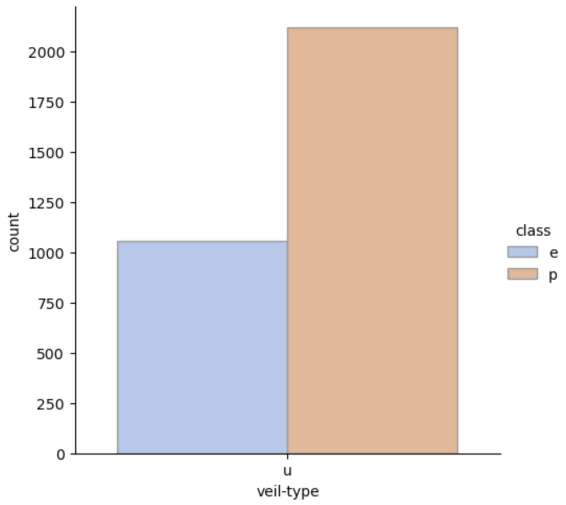


Figure 33: Distribution of veil-type. There are too many missing data in veil-type.

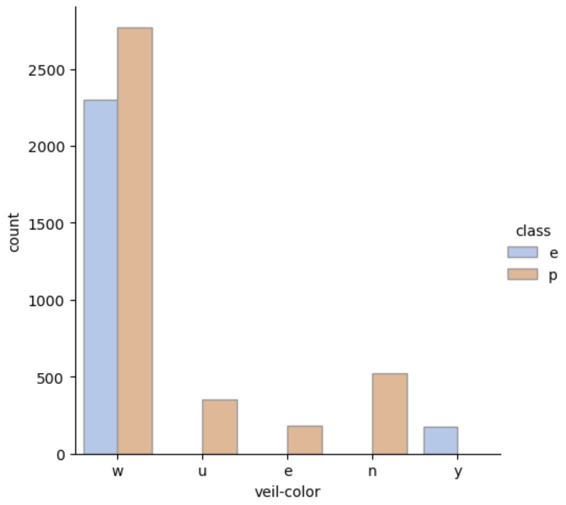


Figure 34: Distribution of veil-color. If veil-color is u ,e or n, it is likely that the mushroom is poisonous. If veil-color is y, it is likely that the mushroom is edible.

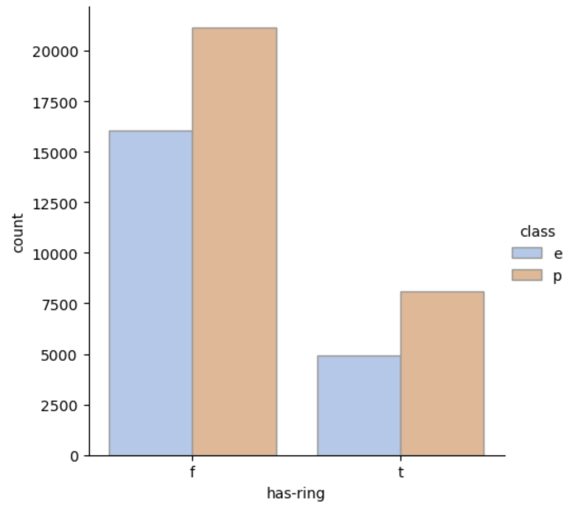


Figure 35: Distribution of has-ring. No distinction.

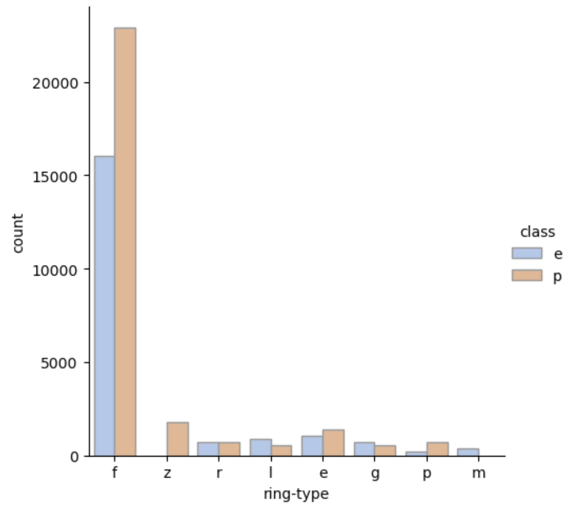


Figure 36: Distribution of ring-type. If ring-type is z, it is likely that the mushroom is poisonous.

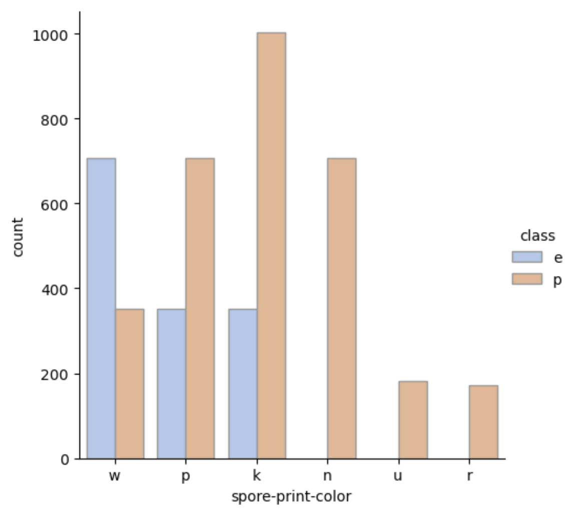


Figure 37: Distribution of spore-print-color. If the spore-print-color is n, u, r. It is likely that the mushroom is poisonous.

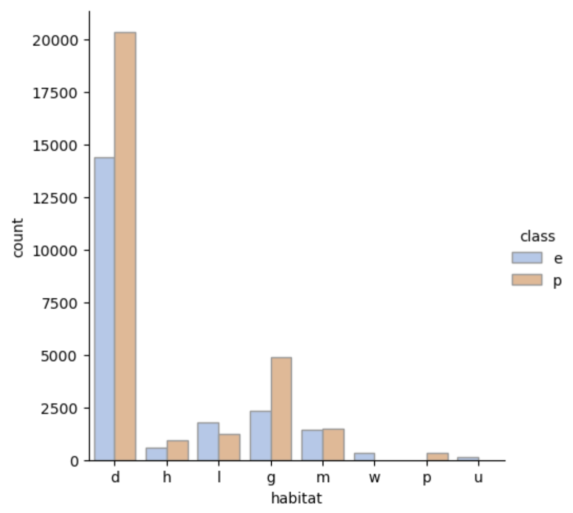


Figure 38: Distribution of habitat. If the habitat is d,p or g. It is likely that the mushroom is poisonous. If the habitat is w or u, it is likely that the mushroom is edible.

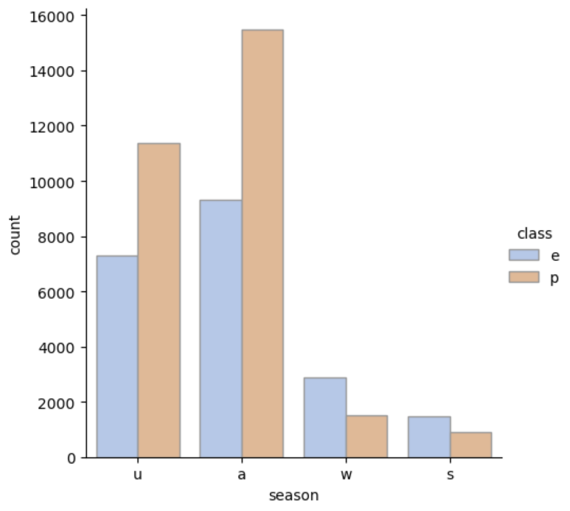
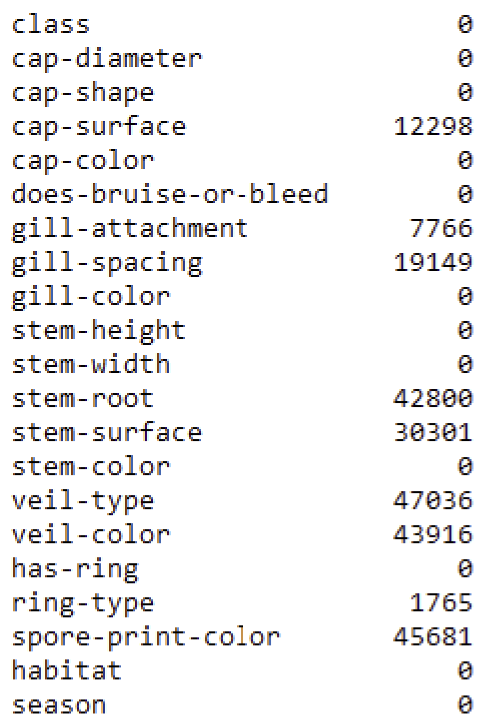


Figure 39: Distribution of season. No distinction.

- In brief, we can not see distinctions between class in the following features:
  - does-bruise-or-bleed
  - Veil-type (will be again in the next part)
  - Has ring
  - Season
- Null values are discussed and handled:

  The first and easy method is to drop columns that contain NaN values. However, they can be useful for predicting specific classes, therefore NaN values can be placed with ‘?’ which means missing. We need to check which methods give the best cross validation score. We can also drop NaN values after One-Hot-Encoding fully utilize the dataset.



```
class                      0
cap-diameter               0
cap-shape                  0
cap-surface            12298
cap-color                  0
does-bruise-or-bleed       0
gill-attachment         7766
gill-spacing           19149
gill-color                 0
stem-height                0
stem-width                 0
stem-root              42800
stem-surface           30301
stem-color                 0
veil-type              47036
veil-color             43916
has-ring                   0
ring-type               1765
spore-print-color      45681
habitat                    0
season                     0
```

Figure 40: Number of missing values in the respective columns

Therefore, following features contain null values:

- Cap-surface
- Gill-attachment
- Gill-spacing
- Stem-root
- Stem-surface
- Veil-type
- Veil-color
- Ring-type
- spore-print-color

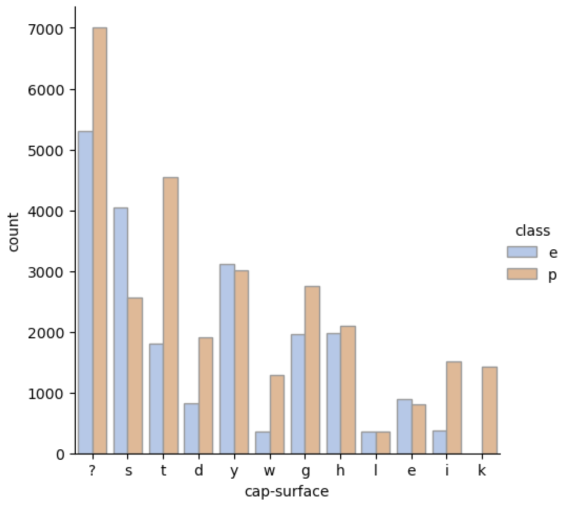


Figure 41: Distribution of cap-surface, NaN values are replaced with ?

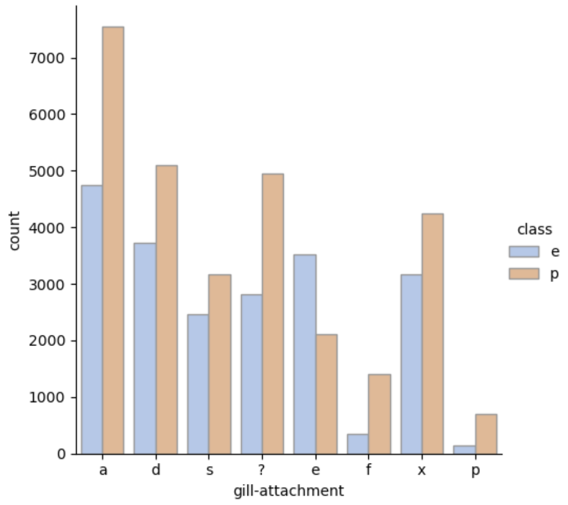


Figure 42: Distribution of gill-attachment, NaN values are replaced with ?

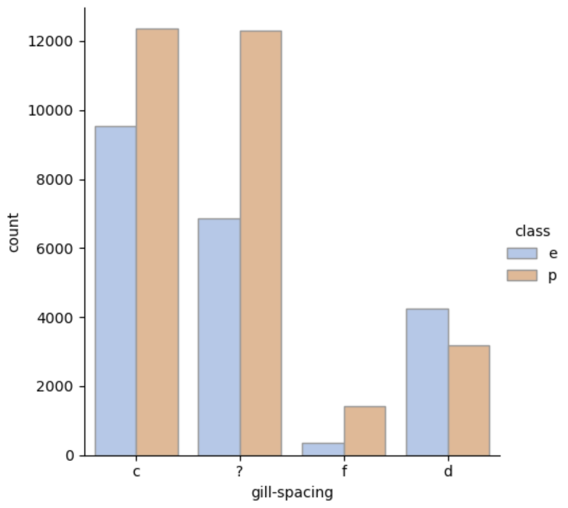


Figure 43: Distribution of gill-spacing, NaN values are replaced with ?

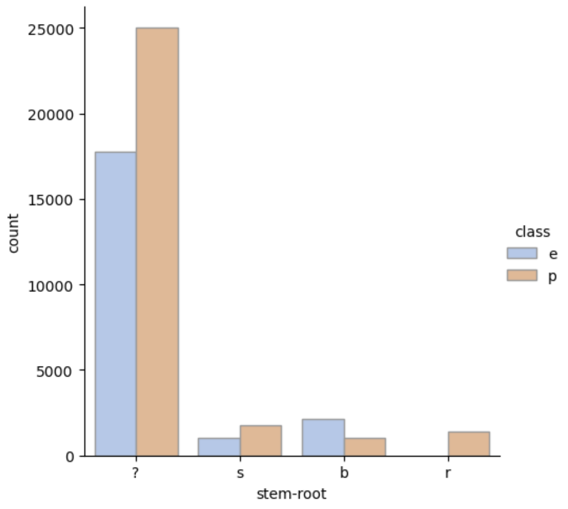


Figure 44: Distribution of stem-root, NaN values are replaced with ?

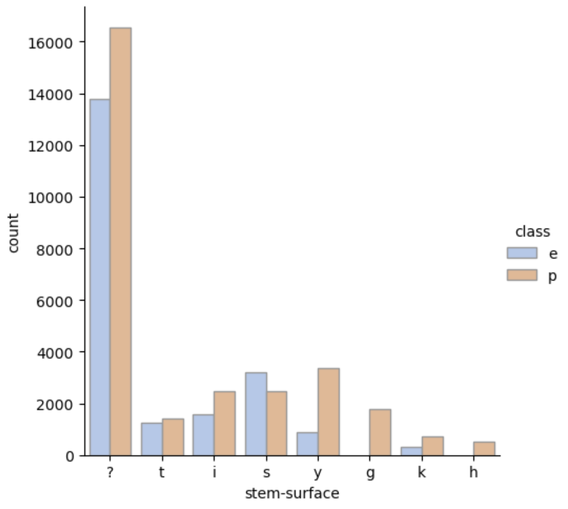


Figure 45: Distribution of stem-surface, NaN values are replaced with ?

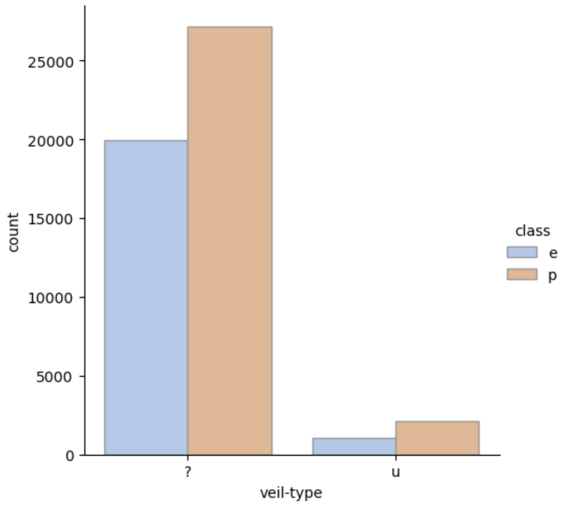


Figure 46: Distribution of veil-type, NaN values are replaced with ?

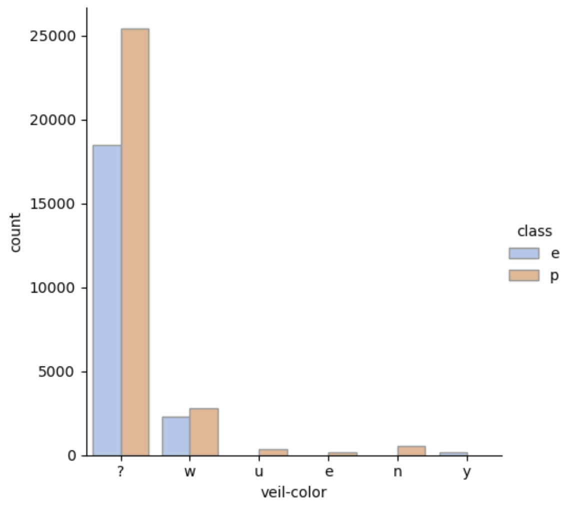


Figure 47: Distribution of veil-color, NaN values are replaced with ?

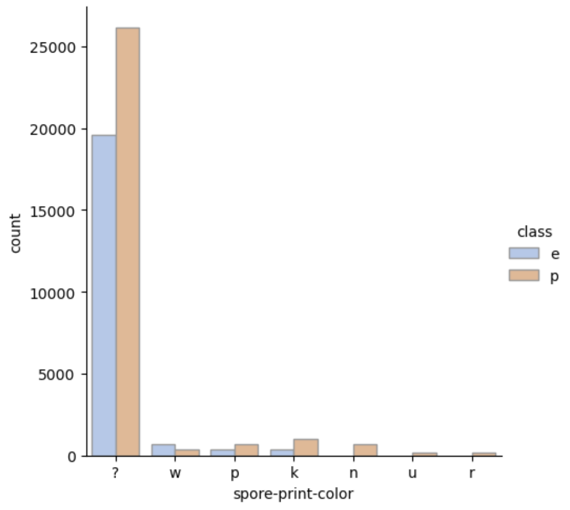


Figure 48: Distribution of spore-print-color, NaN values are replaced with ?

For numerical features, we can also pairplot to see the correlation between numerical features.

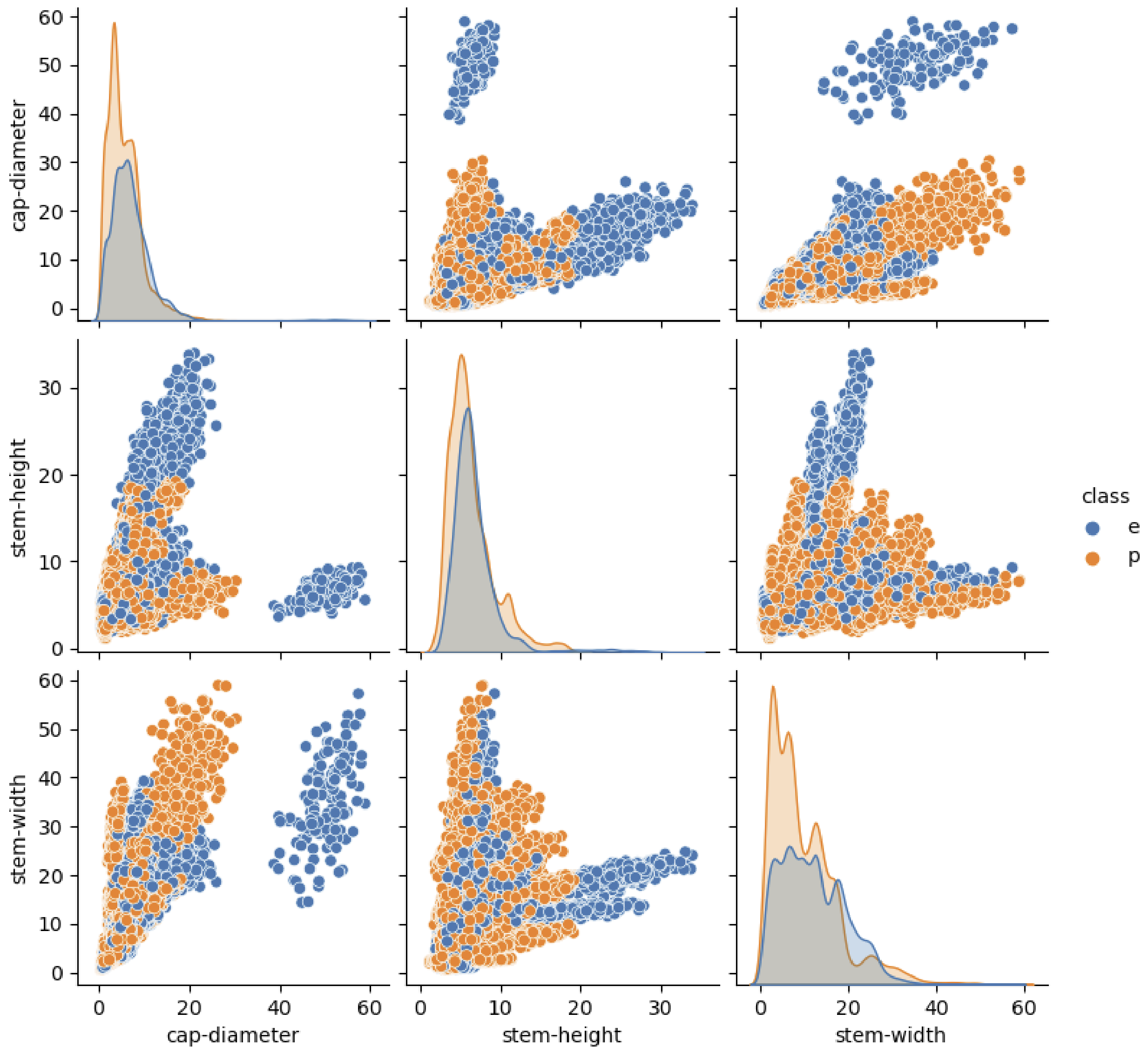


Figure 49: Pair plot of numerical features

Some of the edible samples are linearly separable when we investigate the pair stem-width and cap-diameter, stem-width and stem-height, stem-diameter and stem-height.

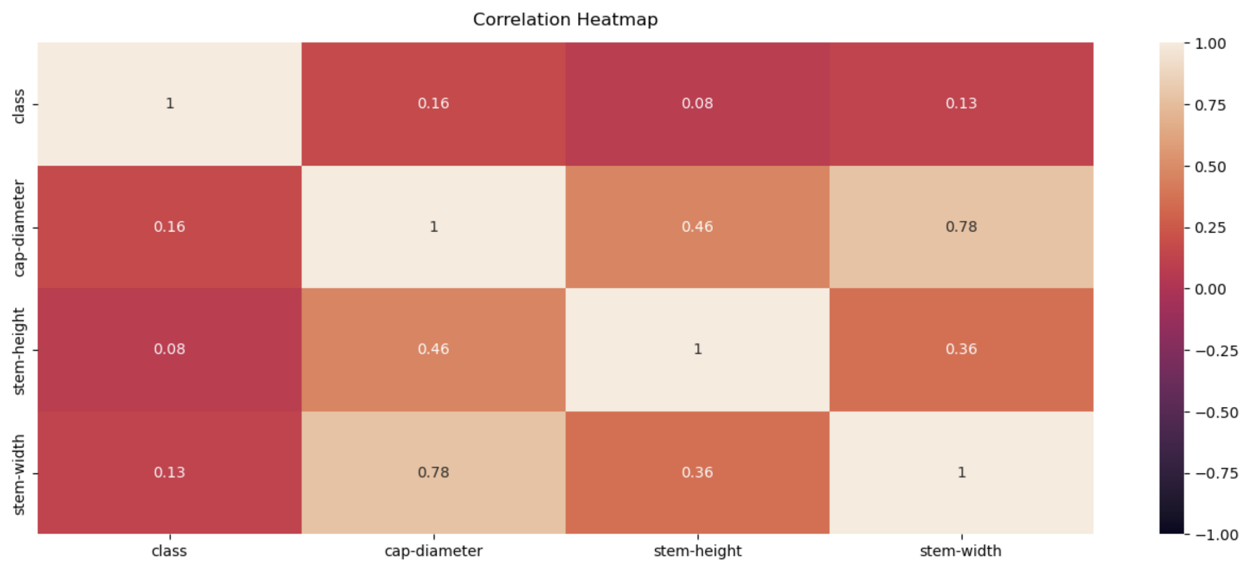


Figure 50: Correlation between numerical features and class.

There is a high correlation between cap-diameter and stem-width.

2) Data Pre-Processing:

Pipeline with StandardScaler for numerical features and OneHotEncoder for categorical features is constructed. I have noticed that the scale of numerical features are not the same and they follow a gaussian distribution, therefore StandardScaler is a good choice for numerical data. Also, OneHotEncoder is a good choice for non-ordinal data. All categorical variables are non-ordinal, therefore using OneHotEncoder is appropriate.

Dataset consists of null values, therefore we should handle null values. I have tried three methods for handling null values.

1) Discarding columns that contain NaN
2) Filling NaN values with ‘?’
3) After one-hot-encoding only columns that have ‘?’ are discarded.

These three methods are compared using cross-validation and logistic regression. For cross-validation k = 10. Model is LogisticRegression.

Data Type	Sample Size	Feature Space Size
Training Data (discarded columns)	50213	62
Testing Data (discarded columns)	10865	62
Training Data (all NaN filled with ?)	50213	122
Testing Data (all NaN filled with ?)	10856	122
Training Data (NaN filled but ? discarded)	50213	113
Testing Data (NaN filled but ? discarded)	10856	113

Table 1: Sample Size and Feature Space Size for processed datasets

Method Type	Validation Accuracy	Validation Precision	Validation Recall	Validation F1
Discard Columns	0.7419	0.7023	0.6635	0.6823
all NaN filled with ?	0.8772	0.8610	0.8419	0.82513
NaN filled but ? discarded	0.764	0.8600	0.8413	0.8505

Table 2: Cross Validation scores for all methods

As it can be easily observed from Table 1 and Table 2, it is wise to use NaN filled but ? discarded since feature space size is less than method 2, and metrics are better than discard columns. Therefore, following parts such as data augmentation uses the NaN filled but ? discarded. Using the third method enables us to nearly use all data given to us. However, our dimension size is huge, therefore feature selection method should be used before model testing.

3) Data Augmentation:

I have added stem-area and stem-cap-size features:

Stem-area = stem-width * stem-height

Stem-cap-size = stem-area *cap-diameter

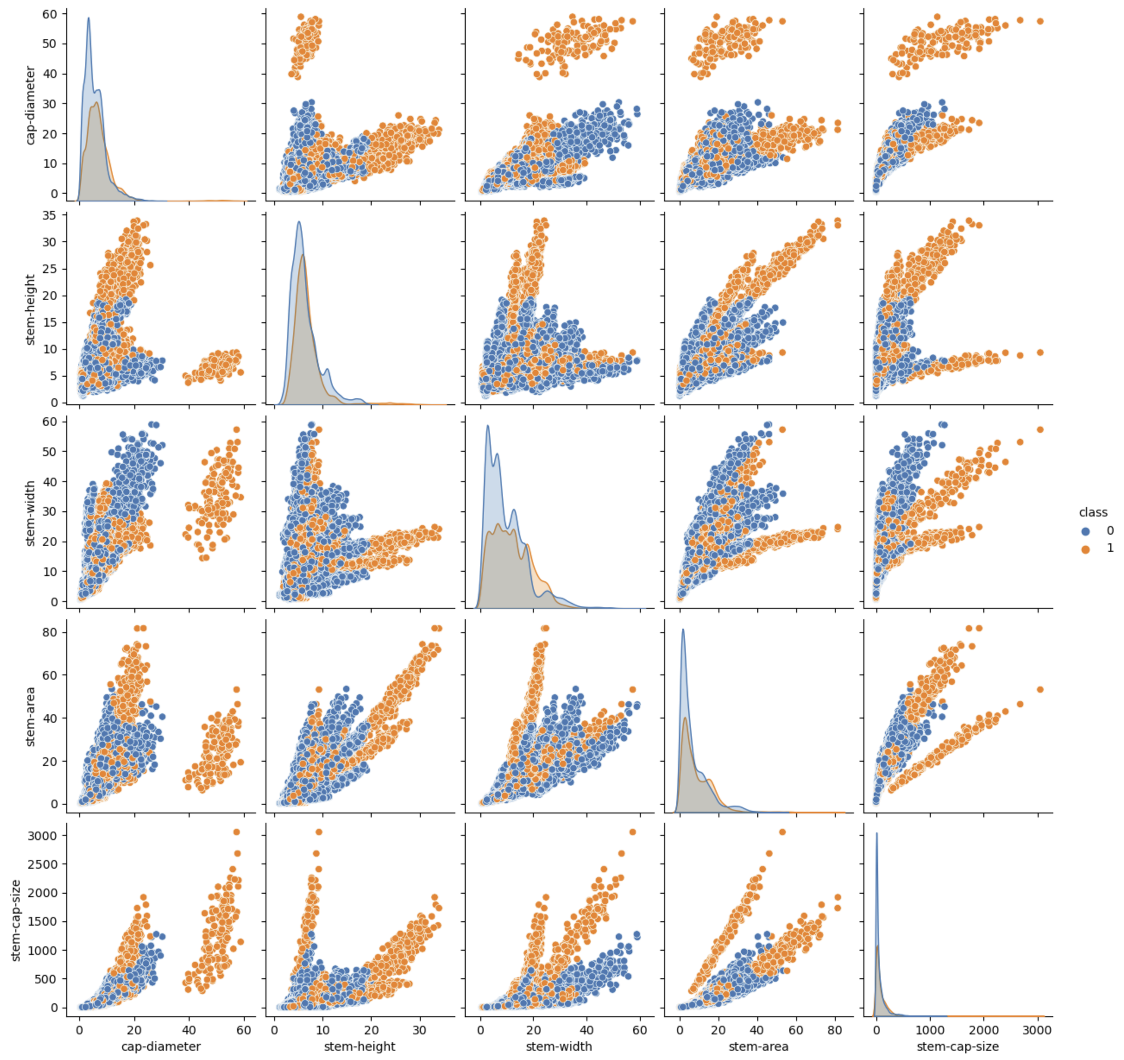


Figure 51: Pairplot of new numerical features

Note that, last 20 plots are related to the new added features. These plots are nearly linearly separable, therefore new features will help to distinguish classes. After adding new features, I have applied the method 3 that is mentioned in the pre-processing and obtained the following size training data.

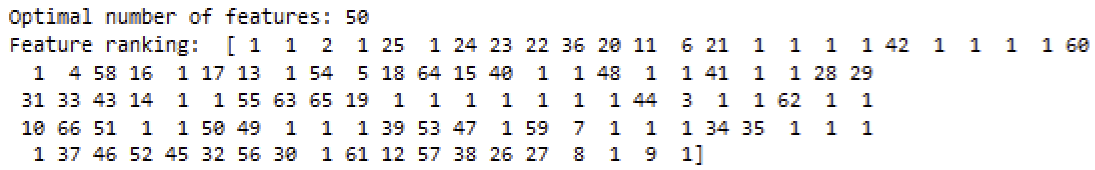
Augmented Training Data Size: 50213 x 115

4) Statistical Hypothesis Testing:

Two analysis are made in this part since hypothesis testing was not giving accurate results when the feature space size is 115.

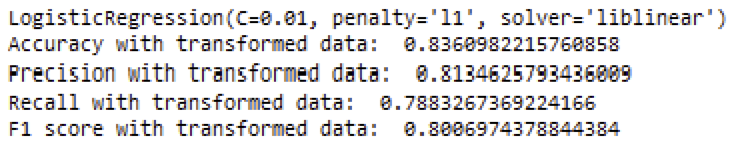
Firstly, feature selection method is used to reduce feature space size to 50.

Before performing hypothesis testing, I have done feature selection with recursive feature elimination selection with cross validation. The goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through any specific attribute or callable. Then, the least important features are pruned from the current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached. I have used LogisticRegression with l1 regularization to select features with not only RFE but also with l1 regularization. RFECV found out 50 distinct features from 115 features.



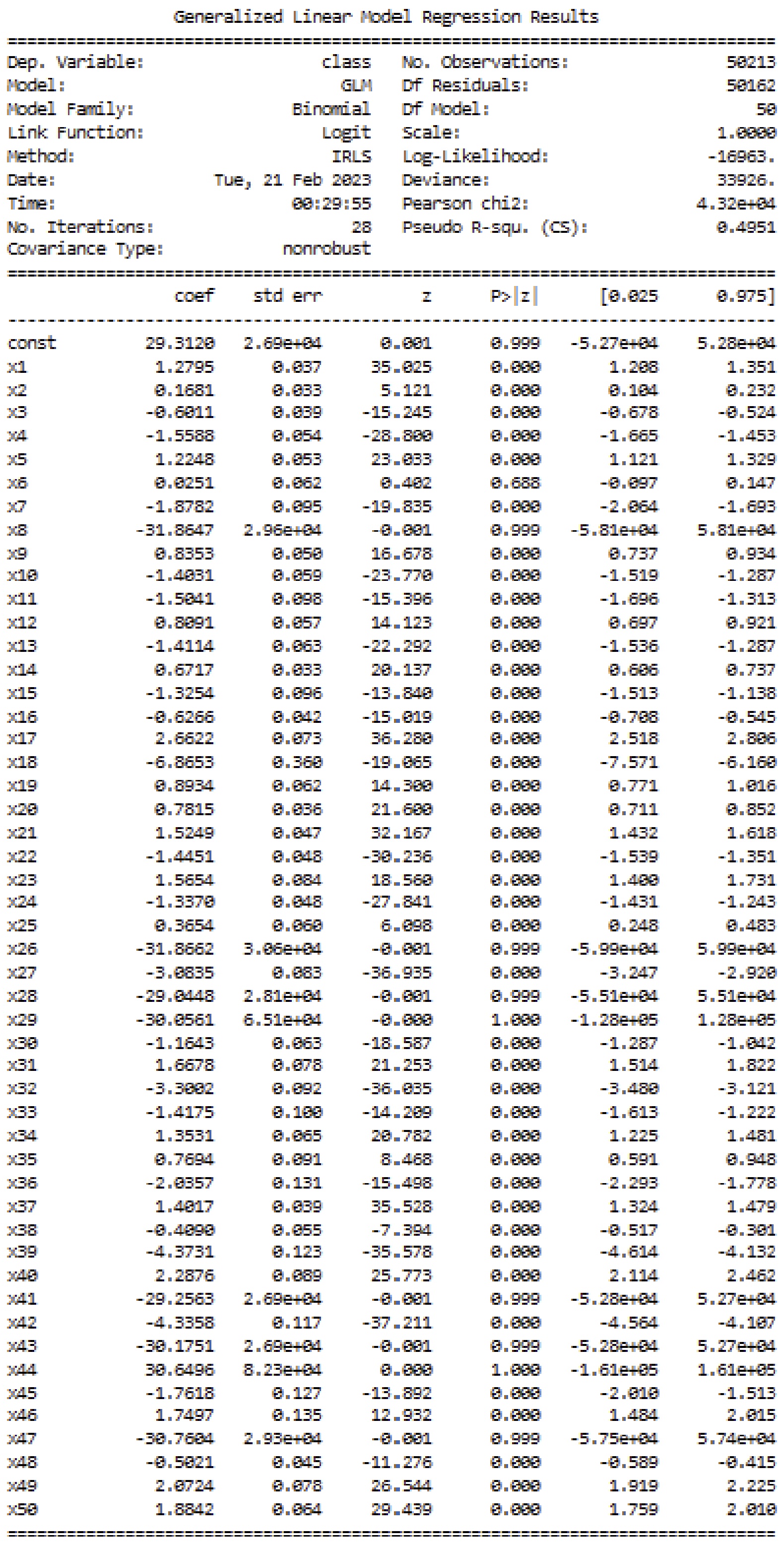
```
Optimal number of features: 50
Feature ranking:  [ 1  1  2  1 25  1 24 23 22 36 20 11  6 21  1  1  1  1 42  1  1  1  1 60
  1  4 58 16  1 17 13  1 54  5 18 64 15 40  1  1 48  1  1 41  1  1 28 29
 31 33 43 14  1  1 55 63 65 19  1  1  1  1  1  1  1 44  3  1  1 62  1  1
 10 66 51  1  1 50 49  1  1  1 39 53 47  1 59  7  1  1  1 34 35  1  1  1
  1 37 46 52 45 32 56 30  1 61 12 57 38 26 27  8  1  9  1]
```

My validation accuracy with the new features was:



```
LogisticRegression(C=0.01, penalty='l1', solver='liblinear')
Accuracy with transformed data:  0.8360982215760858
Precision with transformed data:  0.8134625793436009
Recall with transformed data:  0.7883267369224166
F1 score with transformed data:  0.8006974378844384
```

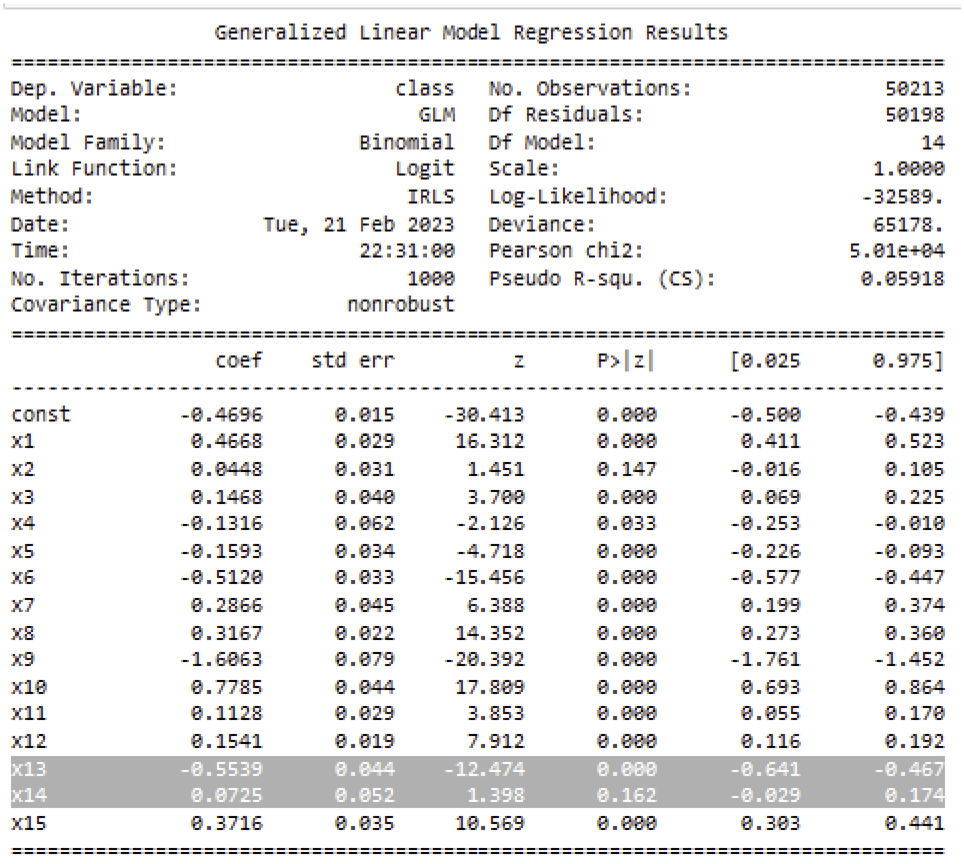
Therefore, the model was not underfitting with low number of features.Then, I was able to continue with statistical hypothesis testing. Hypothesis testing was very difficult without feature selection since features were very imbalanced, therefore the algorithm could not converge to a minimum. Some coefficients standard errors are really high.



```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  class   No. Observations:                50213
Model:                            GLM   Df Residuals:                    50162
Model Family:                Binomial   Df Model:                           50
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -16963.
Date:                Tue, 21 Feb 2023   Deviance:                       33926.
Time:                        00:29:55   Pearson chi2:                 4.32e+04
No. Iterations:                    28   Pseudo R-squ. (CS):             0.4951
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         29.3120   2.69e+04      0.001      0.999   -5.27e+04    5.28e+04
x1             1.2795      0.037     35.025      0.000       1.208       1.351
x2             0.1681      0.033      5.121      0.000       0.104       0.232
x3            -0.6011      0.039    -15.245      0.000      -0.678      -0.524
x4            -1.5588      0.054    -28.800      0.000      -1.665      -1.453
x5             1.2248      0.053     23.033      0.000       1.121       1.329
x6             0.0251      0.062      0.402      0.688      -0.097       0.147
x7            -1.8782      0.095    -19.835      0.000      -2.064      -1.693
x8           -31.8647   2.96e+04     -0.001      0.999   -5.81e+04    5.81e+04
x9             0.8353      0.050     16.678      0.000       0.737       0.934
x10           -1.4031      0.059    -23.770      0.000      -1.519      -1.287
x11           -1.5041      0.098    -15.396      0.000      -1.696      -1.313
x12            0.8091      0.057     14.123      0.000       0.697       0.921
x13           -1.4114      0.063    -22.292      0.000      -1.536      -1.287
x14            0.6717      0.033     20.137      0.000       0.606       0.737
x15           -1.3254      0.096    -13.840      0.000      -1.513      -1.138
x16           -0.6266      0.042    -15.019      0.000      -0.708      -0.545
x17            2.6622      0.073     36.280      0.000       2.518       2.806
x18           -6.8653      0.360    -19.065      0.000      -7.571      -6.160
x19            0.8934      0.062     14.300      0.000       0.771       1.016
x20            0.7815      0.036     21.600      0.000       0.711       0.852
x21            1.5249      0.047     32.167      0.000       1.432       1.618
x22           -1.4451      0.048    -30.236      0.000      -1.539      -1.351
x23            1.5654      0.084     18.560      0.000       1.400       1.731
x24           -1.3370      0.048    -27.841      0.000      -1.431      -1.243
x25            0.3654      0.060      6.098      0.000       0.248       0.483
x26          -31.8662   3.06e+04     -0.001      0.999   -5.99e+04    5.99e+04
x27           -3.0835      0.083    -36.935      0.000      -3.247      -2.920
x28          -29.0448   2.81e+04     -0.001      0.999   -5.51e+04    5.51e+04
x29          -30.0561   6.51e+04     -0.000      1.000   -1.28e+05    1.28e+05
x30           -1.1643      0.063    -18.587      0.000      -1.287      -1.042
x31            1.6678      0.078     21.253      0.000       1.514       1.822
x32           -3.3002      0.092    -36.035      0.000      -3.480      -3.121
x33           -1.4175      0.100    -14.209      0.000      -1.613      -1.222
x34            1.3531      0.065     20.782      0.000       1.225       1.481
x35            0.7694      0.091      8.468      0.000       0.591       0.948
x36           -2.0357      0.131    -15.498      0.000      -2.293      -1.778
x37            1.4017      0.039     35.528      0.000       1.324       1.479
x38           -0.4090      0.055     -7.394      0.000      -0.517      -0.301
x39           -4.3731      0.123    -35.578      0.000      -4.614      -4.132
x40            2.2876      0.089     25.773      0.000       2.114       2.462
x41          -29.2563   2.69e+04     -0.001      0.999   -5.28e+04    5.27e+04
x42           -4.3358      0.117    -37.211      0.000      -4.564      -4.107
x43          -30.1751   2.69e+04     -0.001      0.999   -5.28e+04    5.27e+04
x44           30.6496   8.23e+04      0.000      1.000   -1.61e+05    1.61e+05
x45           -1.7618      0.127    -13.892      0.000      -2.010      -1.513
x46            1.7497      0.135     12.932      0.000       1.484       2.015
x47          -30.7604   2.93e+04     -0.001      0.999   -5.75e+04    5.74e+04
x48           -0.5021      0.045    -11.276      0.000      -0.589      -0.415
x49            2.0724      0.078     26.544      0.000       1.919       2.225
x50            1.8842      0.064     29.439      0.000       1.759       2.010
==============================================================================
```

Figure 52: Results of Hypothesis testing

Numerical features are labeled from x1 to x3, therefore numerical values are statistically significant. However, some p values are found to be 1. These features were misleading our algorithm, therefore they should be discarded. I have learnt that features are very imbalanced and cause logistic regression to fail. Therefore, ensemble methods will probably outperform the standard models. Although the algorithm was able to converge after feature selection, standard errors of some coefficients were too high due to complexity. Then, I applied hypothesis testing on the first 15 features and obtained logical results. Note that, I have selected from 115 features, not from the selected features list by RFECV.



```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  class   No. Observations:                50213
Model:                            GLM   Df Residuals:                    50198
Model Family:                Binomial   Df Model:                           14
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -32589.
Date:                Tue, 21 Feb 2023   Deviance:                       65178.
Time:                        22:31:00   Pearson chi2:                 5.01e+04
No. Iterations:                  1000   Pseudo R-squ. (CS):            0.05918
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.4696      0.015    -30.413      0.000      -0.500      -0.439
x1             0.4668      0.029     16.312      0.000       0.411       0.523
x2             0.0448      0.031      1.451      0.147      -0.016       0.105
x3             0.1468      0.040      3.700      0.000       0.069       0.225
x4            -0.1316      0.062     -2.126      0.033      -0.253      -0.010
x5            -0.1593      0.034     -4.718      0.000      -0.226      -0.093
x6            -0.5120      0.033    -15.456      0.000      -0.577      -0.447
x7             0.2866      0.045      6.388      0.000       0.199       0.374
x8             0.3167      0.022     14.352      0.000       0.273       0.360
x9            -1.6063      0.079    -20.392      0.000      -1.761      -1.452
x10            0.7785      0.044     17.809      0.000       0.693       0.864
x11            0.1128      0.029      3.853      0.000       0.055       0.170
x12            0.1541      0.019      7.912      0.000       0.116       0.192
x13           -0.5539      0.044    -12.474      0.000      -0.641      -0.467
x14            0.0725      0.052      1.398      0.162      -0.029       0.174
x15            0.3716      0.035     10.569      0.000       0.303       0.441
==============================================================================
```

Figure 53: Results of Hypothesis testing

X3 and x14 are found out to be statistically insignificant. These features are stem-width and cap-surface_h. I have also concluded that I will use all the features for model validation.

Dimensionality Reduction using PCA:

We were asked to perform dimensionality reduction using PCA. However, since our variables are categorical, it is not wise to use PCA for the data.

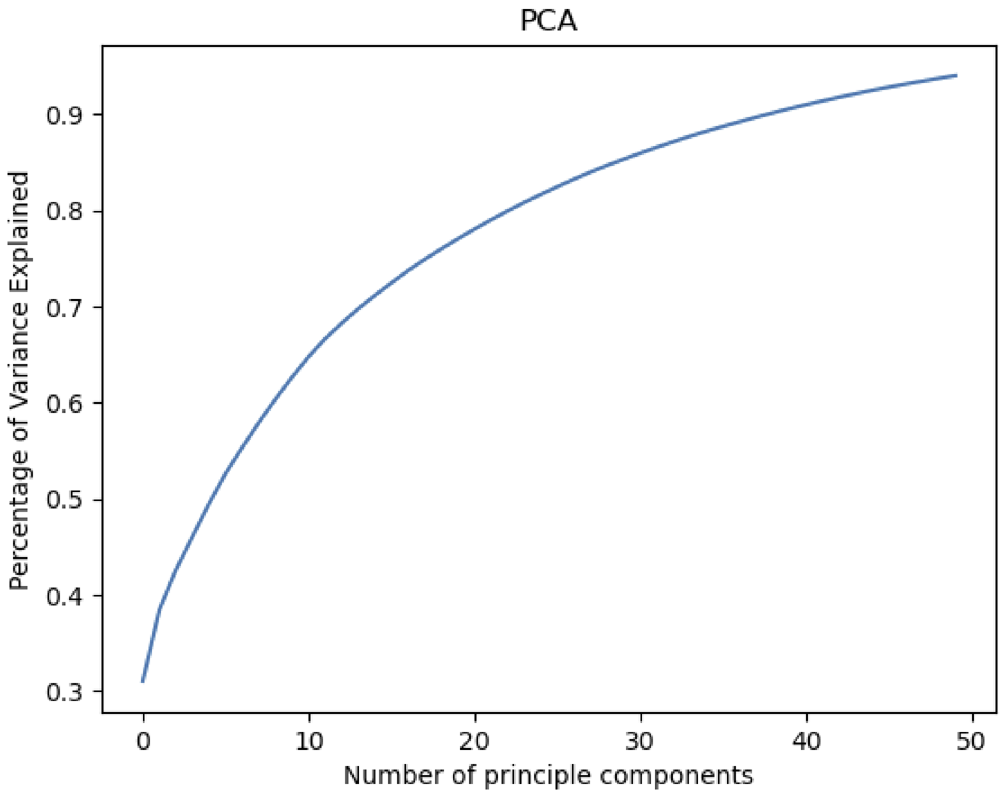


Figure 54: Number of principal components and percentage of variance explained

We were able to explain the 90% variance of the data with only 40 components.

5) **Models of your choice:**

I have chosen LogisticRegression and DecisionTree models for classification. Firstly, they both fit binary classification problems. Secondly, they are good at handling categorical variables. Decision tree is preferred when the number of categorical variables is high. I wanted to both see the performance of a parametric and non-parametric method for this problem. I have also observed that it is easy to overfit decision trees, therefore hyperparameter selection is important for both methods. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem. Logistic regression is preferred when the data is linearly separable. I have observed that data was linearly separable for numerical features. The downside of the logistic regression is that it might fail to grasp the nonlinearity caused by the categorical variables. Logistic regression was also not overfitting to the data or hyperparameters.

Cross validation results for SVM, LogisticRegression and Decision Tree: I have also tried to see the performance of SVM.

Model	Validation accuracy	Validation precision	Validation recall	Validation f1 score
SVM	1	1	1	1
Log Reg	0.8918	0.8797	0.8584	0.8689
Decision Tree	0.9993	0.9992	0.9990	0.9991

Table 3: 10-fold cross validation scores of models

SVM model was SVM(kernel = rbf)

Logistic regression model was LogisticRegression(penalty = ‘l1’,solver=’liblinear’)

DecisionTreeClassifier was the default classifier.

6) Ensemble Method:

I have chosen RandomForestClassifier for the ensemble method part. Random forests can overcome the problem of overfitting decision trees with ensembling. Random forests are also suitable for binary classification and uses averaging to improve the predictive accuracy and control over-fitting. Since we have too many categorical variables, using decision trees is important. Random forests also utilize the categorical variables.

Cross validation results for Random Forest, GradientBoostingClassifier and AdaBoostClassifier: I have also tried to see the performance of GBC and ABC.However, random forest is the best ensemble method for the problem.

Model	Validation accuracy	Validation precision	Validation recall	Validation f1 score
Random Forest	0.9999	0.9999	1	0.9999
GBC	0.9472	0.9428	0.9301	0.9364
ABC	0.8349	0.8011	0.8046	0.8028

Table 4: 10-fold cross validation scores of models

All models are default models, hyperparameter optimization is done in the next section.

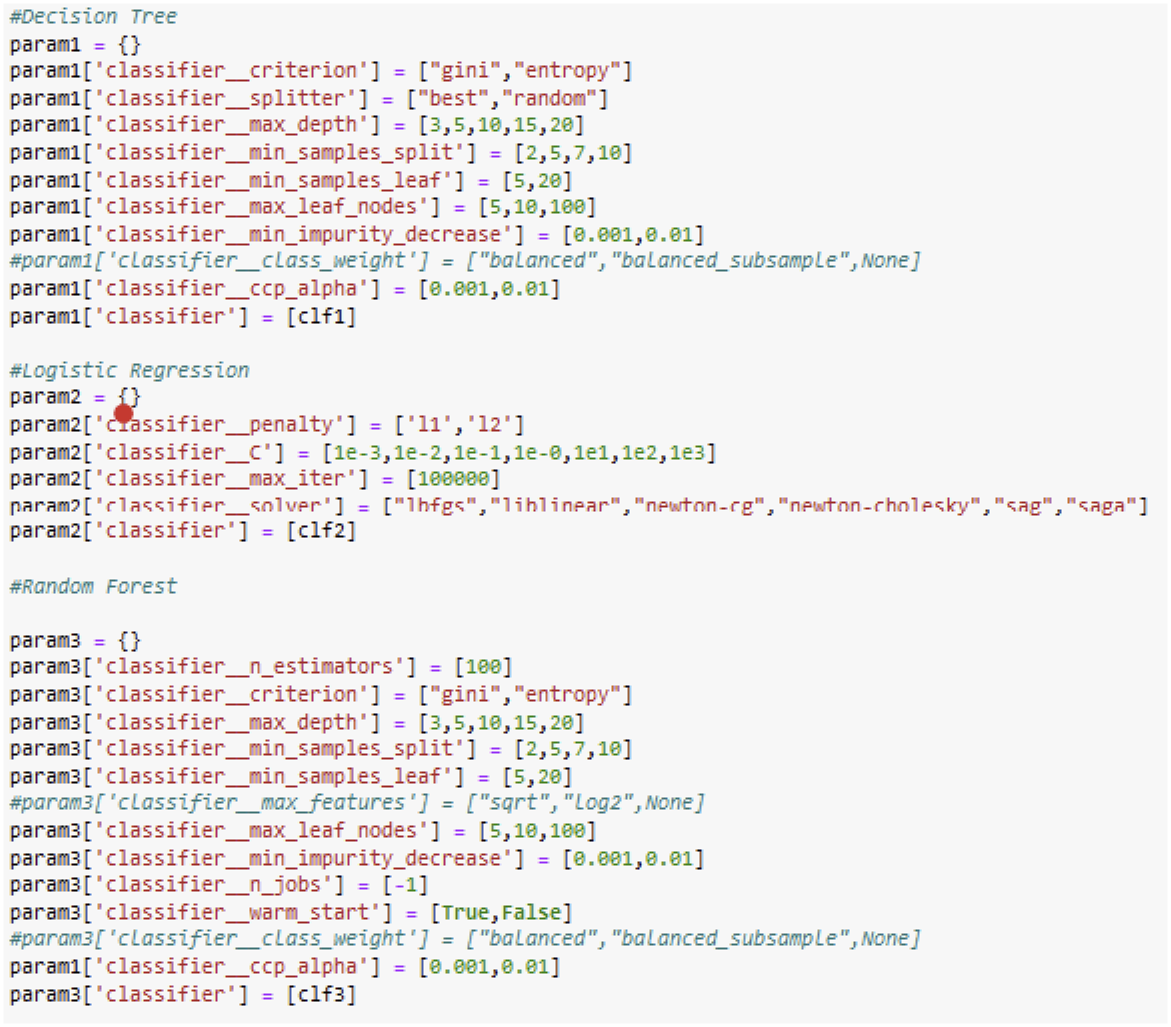
7) Hyperparameter Tuning

I have used the GridSearchCV method to find the best hyperparameters for three models. GridSearchCV takes a classifier and list of input parameters, then using the cross validation technique finds the best combination of input parameters according to the given score. I have chosen f1 score, since recall can be low due to feature imbalance.

Following input list is used for GridSearchCV:

Param1 is for decision tree, param2 is for logistic regression, param3 is for random forest.

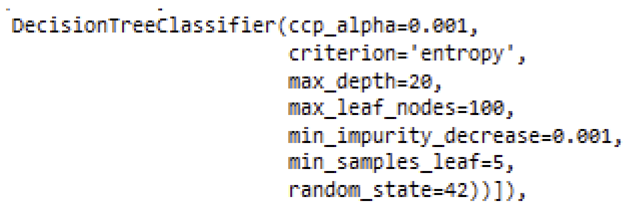
In order to avoid overfitting in tree structured classifier max_depth should be limited, otherwise complexity of the tree will be very high. For logistic regression, it is important to select the regularization parameter to generalize well. Ccp_alpha is used for pruning which is also a important hyperparameter for decision trees.



```
#Decision Tree
param1 = {}
param1['classifier__criterion'] = ["gini","entropy"]
param1['classifier__splitter'] = ["best","random"]
param1['classifier__max_depth'] = [3,5,10,15,20]
param1['classifier__min_samples_split'] = [2,5,7,10]
param1['classifier__min_samples_leaf'] = [5,20]
param1['classifier__max_leaf_nodes'] = [5,10,100]
param1['classifier__min_impurity_decrease'] = [0.001,0.01]
#param1['classifier__class_weight'] = ["balanced","balanced_subsample",None]
param1['classifier__ccp_alpha'] = [0.001,0.01]
param1['classifier'] = [clf1]

#Logistic Regression
param2 = {}
param2['classifier__penalty'] = ['l1','l2']
param2['classifier__C'] = [1e-3,1e-2,1e-1,1e-0,1e1,1e2,1e3]
param2['classifier__max_iter'] = [100000]
param2['classifier__solver'] = ["lbfgs","liblinear","newton-cg","newton-cholesky","sag","saga"]
param2['classifier'] = [clf2]

#Random Forest

param3 = {}
param3['classifier__n_estimators'] = [100]
param3['classifier__criterion'] = ["gini","entropy"]
param3['classifier__max_depth'] = [3,5,10,15,20]
param3['classifier__min_samples_split'] = [2,5,7,10]
param3['classifier__min_samples_leaf'] = [5,20]
#param3['classifier__max_features'] = ["sqrt","log2",None]
param3['classifier__max_leaf_nodes'] = [5,10,100]
param3['classifier__min_impurity_decrease'] = [0.001,0.01]
param3['classifier__n_jobs'] = [-1]
param3['classifier__warm_start'] = [True,False]
#param3['classifier__class_weight'] = ["balanced","balanced_subsample",None]
param1['classifier__ccp_alpha'] = [0.001,0.01]
param3['classifier'] = [clf3]
```

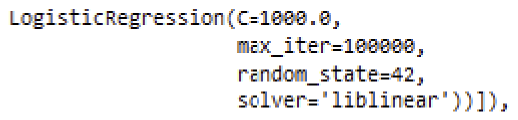
Figure 55: Hyperparameters used for GridSearchCV

Best classifier parameters for decision tree:



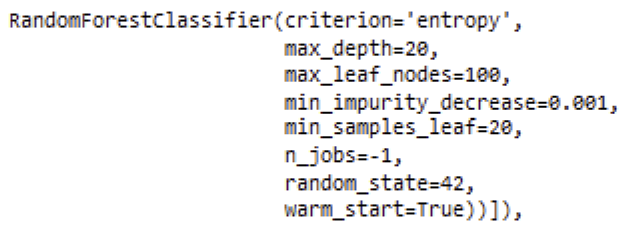
```
DecisionTreeClassifier(ccp_alpha=0.001,
                       criterion='entropy',
                       max_depth=20,
                       max_leaf_nodes=100,
                       min_impurity_decrease=0.001,
                       min_samples_leaf=5,
                       random_state=42))]),
```

Best classifier parameters for LogisticRegression: (l2-regularization)



```
LogisticRegression(C=1000.0,
                   max_iter=100000,
                   random_state=42,
                   solver='liblinear'))]),
```

Best classifier parameters for RandomForestClassifier:



```
RandomForestClassifier(criterion='entropy',
                       max_depth=20,
                       max_leaf_nodes=100,
                       min_impurity_decrease=0.001,
                       min_samples_leaf=20,
                       n_jobs=-1,
                       random_state=42,
                       warm_start=True))]),
```

8) Results:

In this part, I have discussed training metrics and testing metrics.

Model	Training Accuracy	Training Precision	Training Recall	Training F1 Score
Decision Tree	0.9913	**0.9962**	0.9829	0.9895
Logistic Regression	0.8937	0.8815	0.8612	0.8712
Random Forest Classifier	**0.9956**	0.9947	**0.9948**	**0.9947**

Table 5: Training Metrics for three models with the best hyperparameters

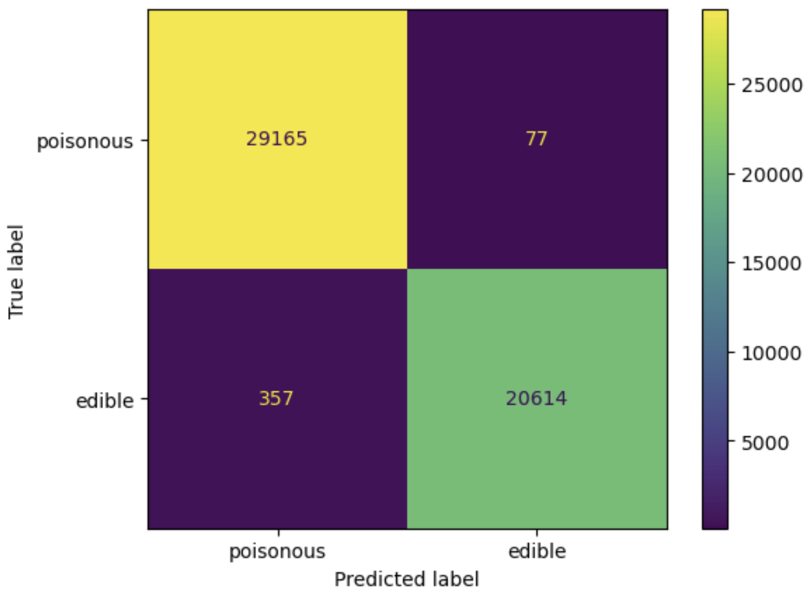


Figure 56: Decision Tree Training Confusion Matrix

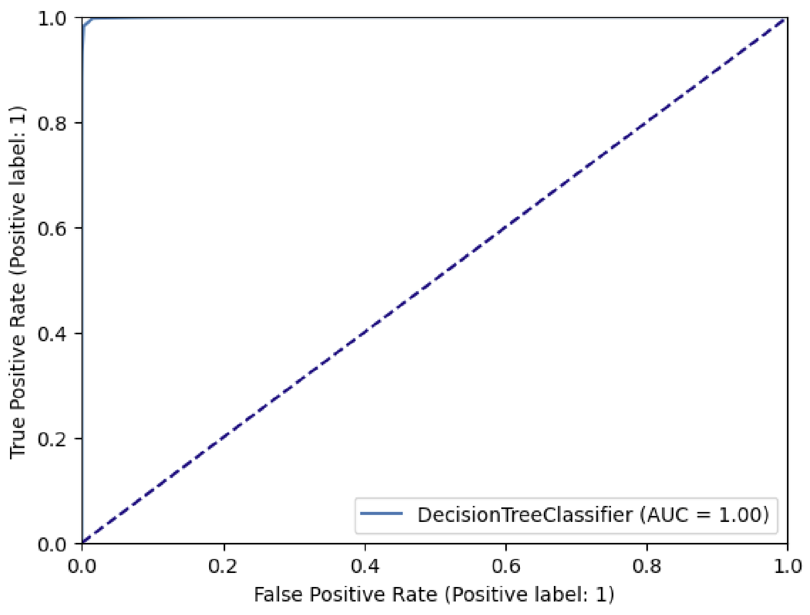


Figure 57: Decision Tree Training ROC Curve and AUC

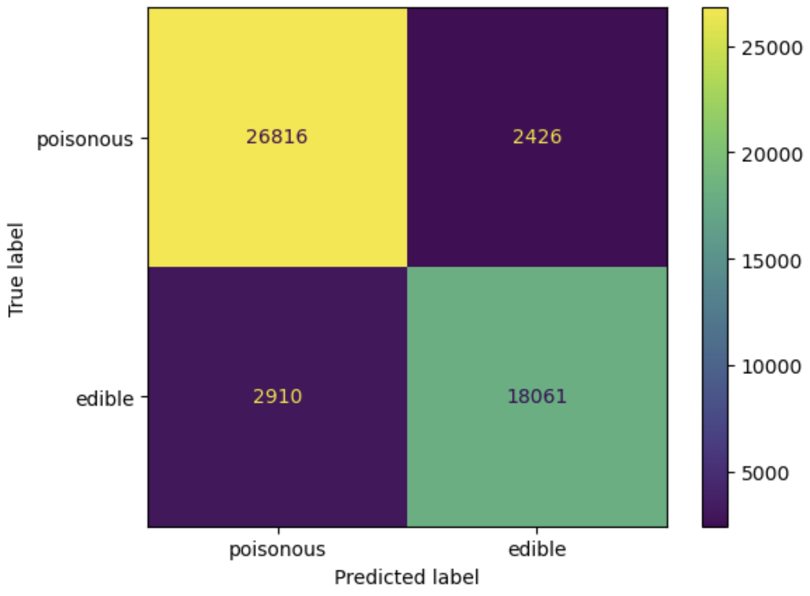


Figure 58: Logistic Regression Training Confusion Matrix

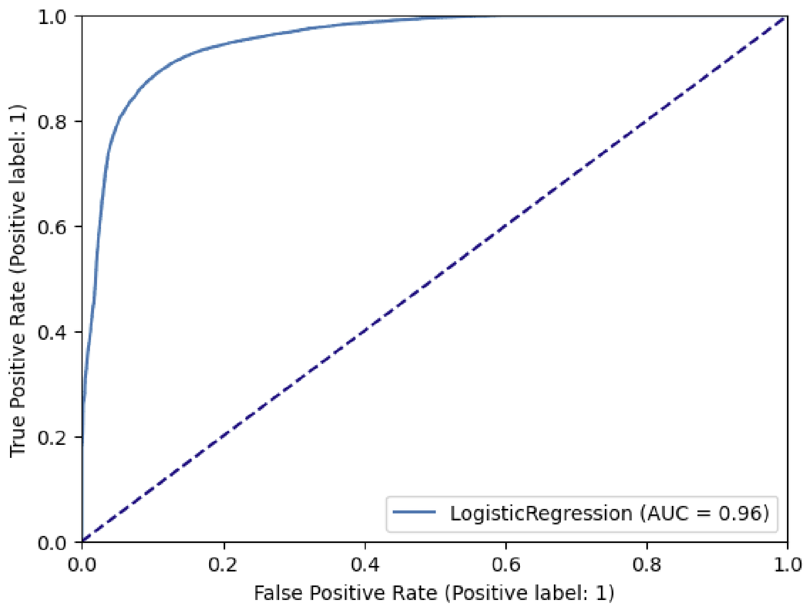


Figure 59: Logistic Regression Training ROC Curve and AUC

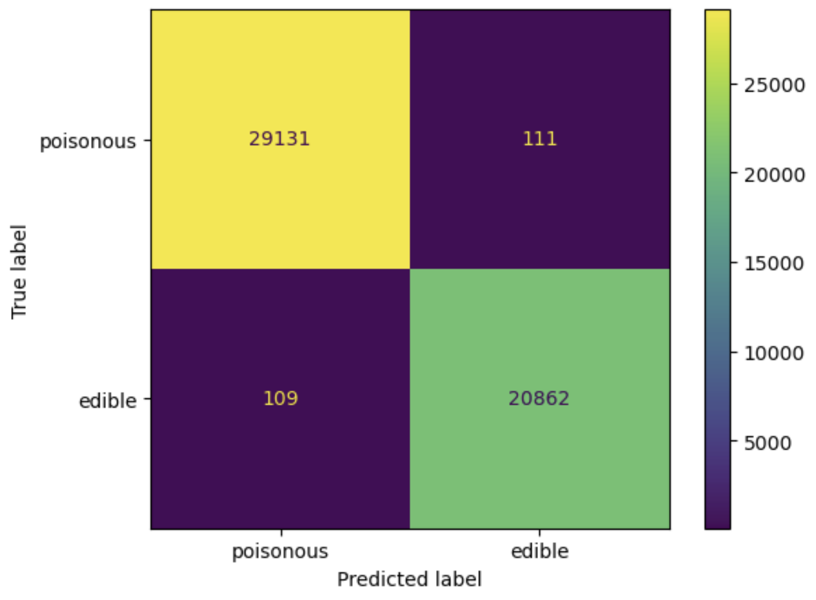


Figure 60: Random Forest Training Confusion Matrix

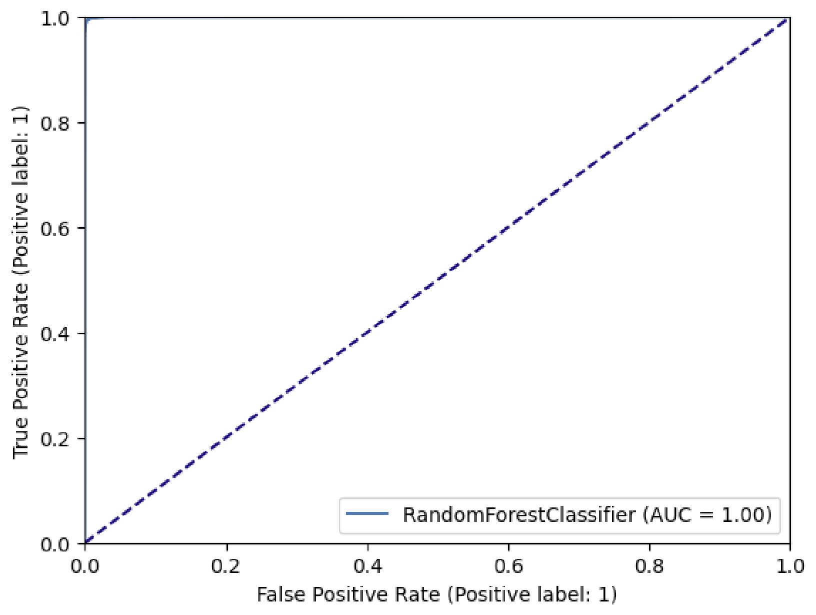


Figure 61: Random Forest Training ROC Curve and AUC

Model	Testing Accuracy	Testing Precision	Testing Recall	Testing F1 Score
Decision Tree	0.5306	0.7073	0.3062	0.4274
Logistic Regression	0.4118	0.4734	0.2510	0.3281
Random Forest Classifier	**0.6042**	**0.8902**	**0.3515**	**0.5040**

Table 6: Testing metrics for three models with the best hyperparameters

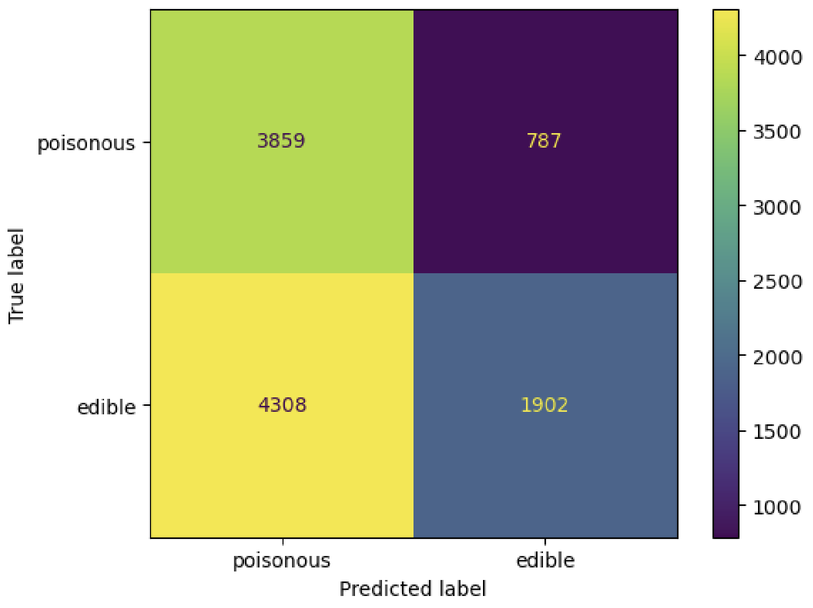


Figure 62: Decision Tree Testing Confusion Matrix

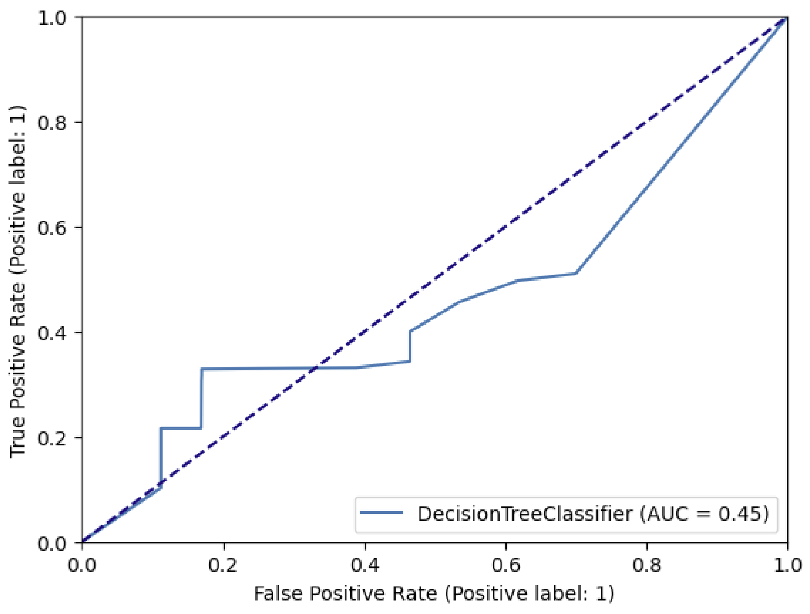


Figure 63: Decision Tree Testing ROC Curve and AUC

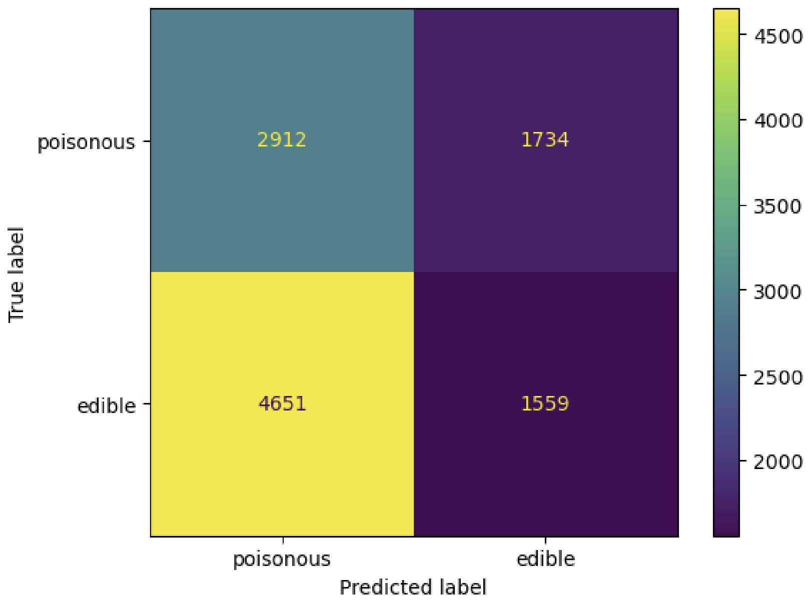


Figure 64: Logistic Regression Testing Confusion Matrix

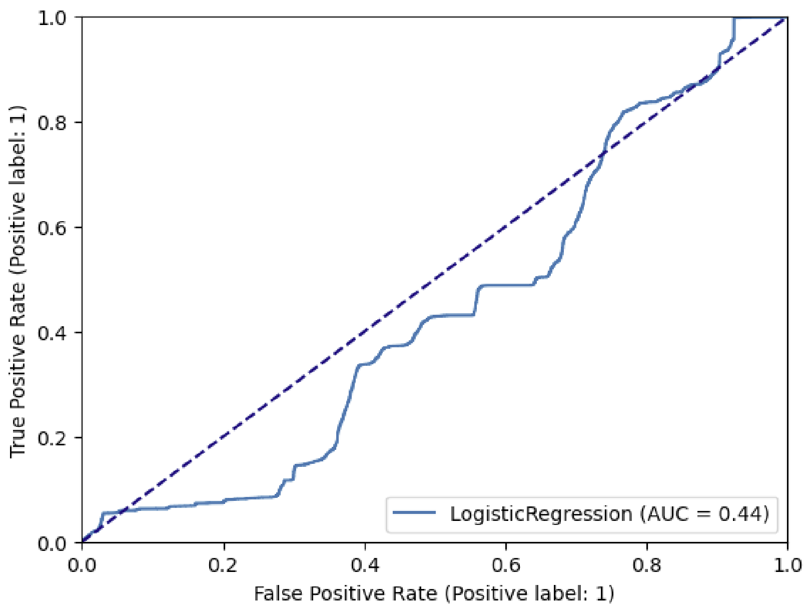


Figure 65: Logistic Regression Testing ROC Curve and AUC

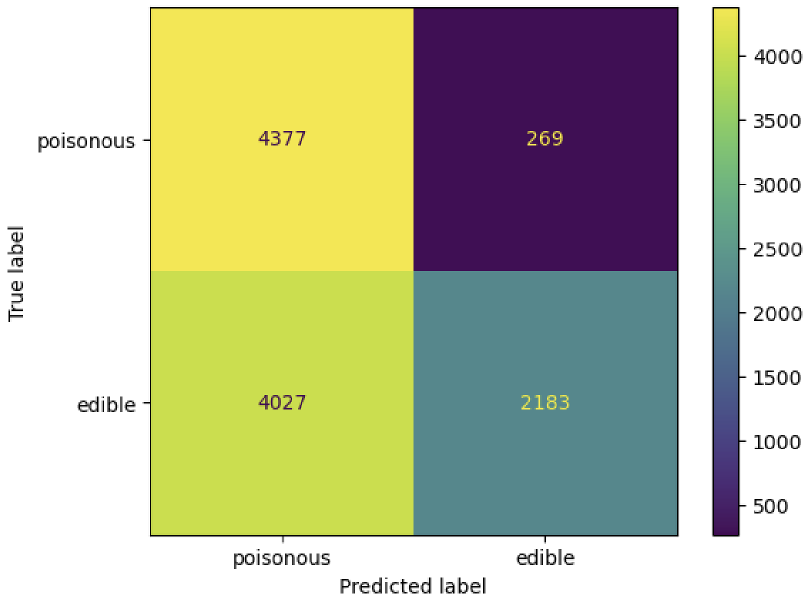


Figure 66: Random Forest Testing Confusion Matrix

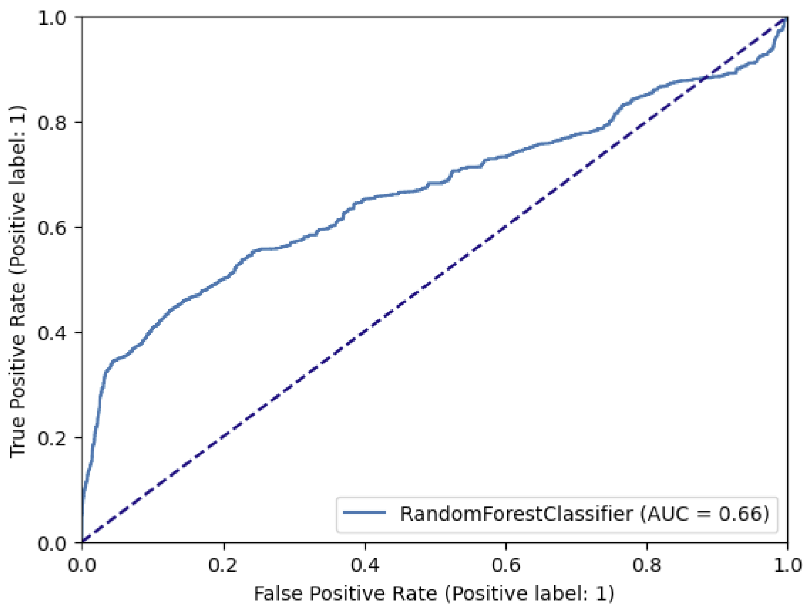


Figure 67: Random Forest Testing ROC Curve and AUC

Cross validation technique: Since we are not final with our model, we can not use testing data to tune hyperparameters. However, a technique called cross validation is used for hyperparameter tuning. Dataset is splitted into k-folds. For each validation step, a fold is assigned as validation and remaining folds are assigned as training dataset. The model is trained using the training dataset and metrics are obtained using the validation dataset. This process continues for k times and average of the metrics are used to determine the best hyperparameter set. Therefore, we were able to choose the best hyperparameter set without using a test dataset.

My training results were very good. My cross-validation results were similar to the training results, therefore I believe that I was able to create a good model for the training dataset. However, the training dataset has many inherited problems which are discussed at the conclusion.

My logistic regression model was performing worse than the random classifier, however I was performing well on the validation dataset. This can be caused by the nonlinearity of the categorical variables. Although the dataset was separable in numerical values, the same was not true in the case of categorical variables. I was able to perform better than the random classifier in the case of decision tree classifier and random forest classifier. The main problem in those classifier's was the recall value. In the data visualization part, it can be easily seen that it is easy to differentiate poisonous mushrooms from all data, however it is hard to differentiate edible mushrooms from all data. All models were yielding low recall value due to high number of false negatives.

Edible samples were predicted as poisonous since edible samples were not carrying distinctive features. Best classifier was the random forest classifier, it performed slightly better than the decision tree classifier, since it is an ensemble method and avoids the overfitting problem. Note that, precision is very high, we are able to detect all poisonous mushrooms which is important than detecting edible mushrooms in our context.

9) Conclusion:

Firstly, null values are replaced with ‘?’. Then, data is preprocessed using StandardScaler and OneHotEncoding. An extra feature naming ‘?’ for each column that has NaN value added to the feature list. Those features are discarded from the feature set. Since hypothesis testing was not converging properly, I have tried to reduce the dimension of the dataset using RFECV technique. The feature space size has reduced to 50. Then, I was able to partially do hypothesis testing. However, due to imbalance in features, hypothesis testing was not giving accurate results, fits were changing too much. Lastly, using the dataset with feature size 50. I have tried to see the performance of three models which are Logistic Regression, Decision Tree and Random Forest.

I have chosen Random Forest to use on my adventure through Mushroomia since it has the best metrics. It has generalized better than the other models. Ensemble methods generalize better than regular methods if the features are uncorrelated.

The main issue I have encountered in this project is the feature imbalance. For example, cap-shape x was observed in nearly 35000 samples, but cap-shape o was only observed in nearly 1500 samples. This was occurring nearly for all categorical values, therefore decision tree models were overfitting due to low number samples in some categories and logistic regression was not producing statistically correct results. In each bootstrap, coefficients were highly fluctuating. Also, multicollinearity was present in the dataset, coefficients were changing in each fit. Another issue, I have observed that edible was not differentiable from all data but, poisonous was differentiable from all data. For example, cap-surface k only contains poisonous samples but edible samples were contained in each category except k. By observing data, I was not able to see any distinctive features of edible mushrooms. Also, correlations between numerical features and the labels were very low. Correlation between cap-diameter and class was 0.16. Other correlations were lower than 0.16. Also, data contains many missing values, especially for stem-root. Using stem-root for validation was effective, but for testing I believe using stem-root yields to overfitting due to the low number of samples for stem-root.

I think that the training dataset is not sampled from the population of the testing dataset since I was able to get high validation metrics but testing metrics were poor.