Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022

Mushroom Classification

For the final project of this class, I choose to build and apply the classification models that we have learned in class to one of the data sets found online. The data set includes multiple entries of mushrooms along with their features. I am going to utilize logistic regression, support vector machine, and random forest to code the models and calculate how accurate they can determine if a mushroom is deadly or safe to eat based on its features. In addition, I am going to use a correlation matrix to find which feature has dominated most in calculating the accuracies of the models.

"Mushroom Classification" is the data set found on a public online community Kaggle for data science and machine learning enthusiasts. There are 23 species of mushrooms taken from The Audubon Society Field Guide to North American Mushrooms (1981). Each mushroom is identified as edible, poisonous, or of unknown edibility. The latter was eventually combined with the poisonous class.

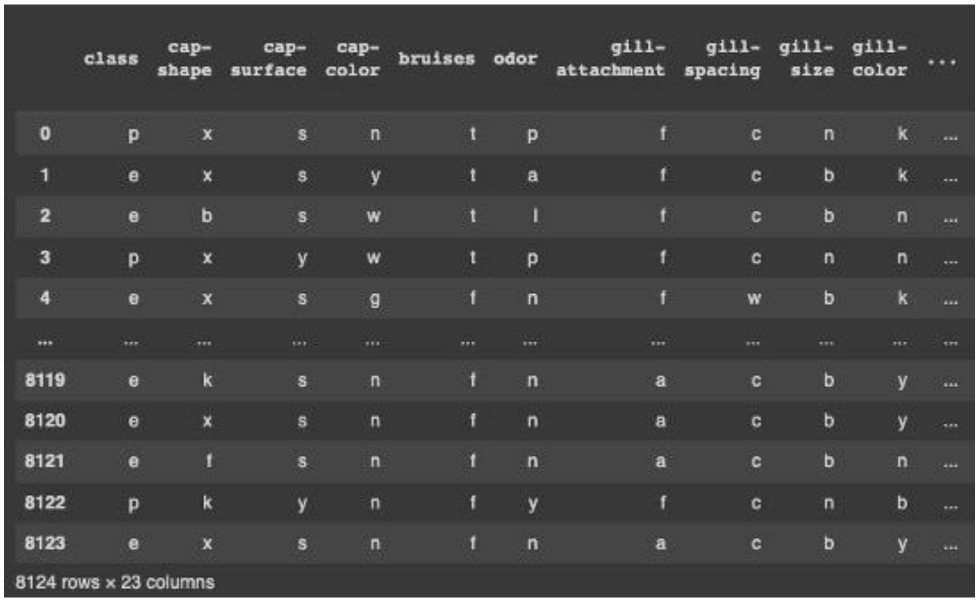
The data set includes 8124 rows and 23 columns. The first column is a class column which indicates if a row is a "p", poisonous, or "e", edible. Each row is a combination of 22 attributions. The data attribution ranges from the mushroom's cap-shape to its habitat. Each attribution includes from 2 to 9 different features to indicate if the mushroom's gill size is "b", broad, or "n", narrow, for example. In Figure 1, we can see that the data set is represented by characters which are the first letters of the features. To work with this data set, I need to preprocess the data first.

Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022



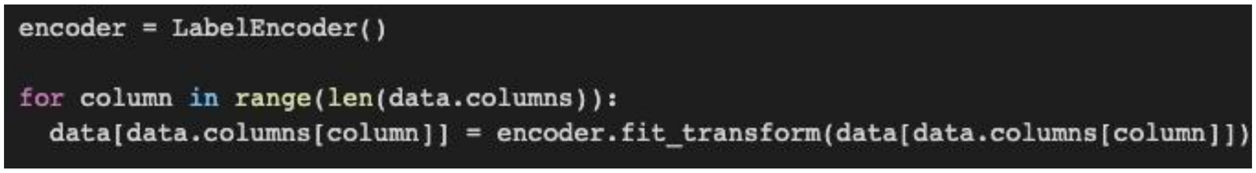
	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...
3	p	x	y	w	t	p	f	c	n	n	...
4	e	x	s	g	f	n	f	w	b	k	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	e	k	s	n	f	n	a	c	b	y	...
8120	e	x	s	n	f	n	a	c	b	y	...
8121	e	f	s	n	f	n	a	c	b	n	...
8122	p	k	y	n	f	y	f	c	n	b	...
8123	e	x	s	n	f	n	a	c	b	y	...

8124 rows × 23 columns

Figure 1

A crop of the mushroom data set

There are two classes I have imported for my project from the ‘sklearn.preprocessing’ package, LabelEncoder and StandardScaler. LabelEncoder is a tool used in machine learning to encode labels with numeric values. This is necessary because most machine learning algorithms require input data to be numerical, not categorical. For example, if we have a dataset with a column containing the strings "cat" and "dog", we can use LabelEncoder to convert these strings into numerical values such as 0 and 1.



```
encoder = LabelEncoder()

for column in range(len(data.columns)):
  data[data.columns[column]] = encoder.fit_transform(data[data.columns[column]])
```

Figure 2

Transforming non-numerical labels to numerical labels

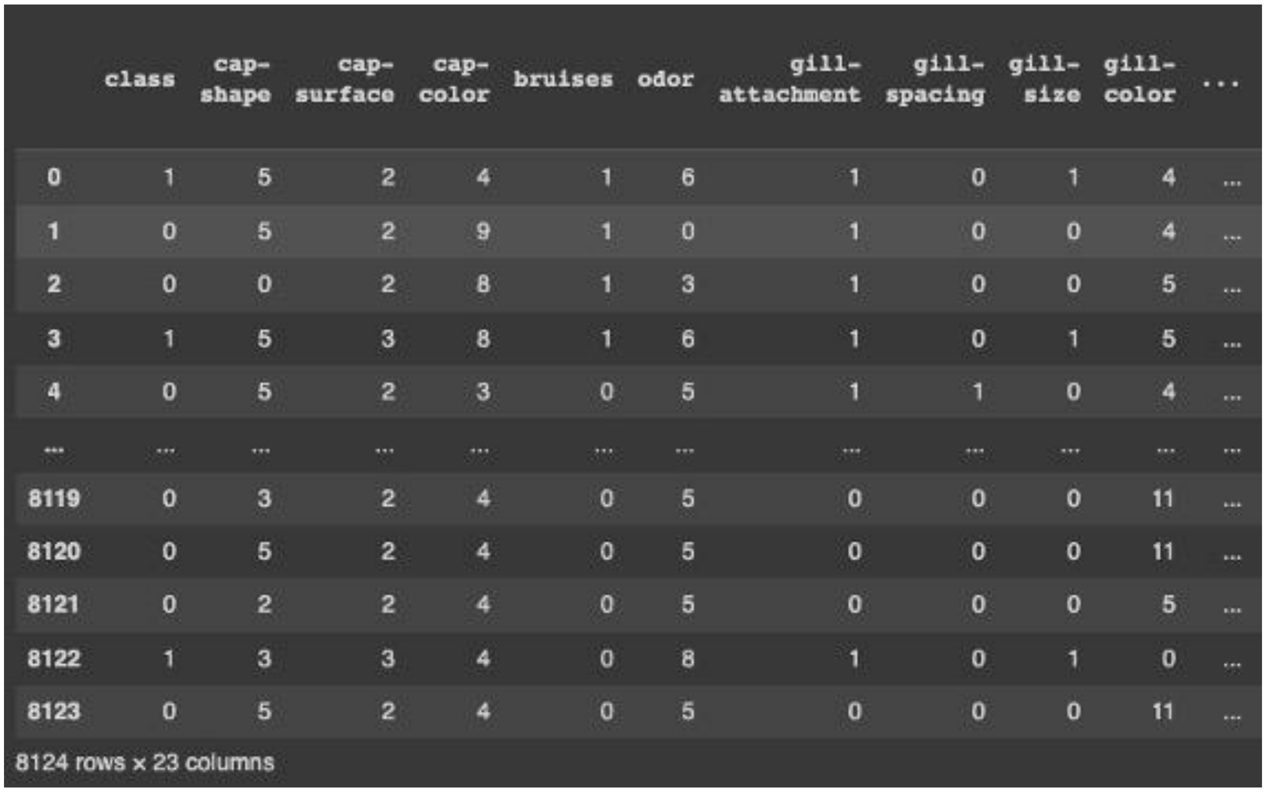
Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022

In Figure 2, I am looping through each column in the data. Then, as I grab the whole slice of the data's column, the instance of the encoder is fit into the data and transforms the labels of that column back into the dataframe. After transforming the labels, they will be represented by numerical values instead of the original strings. Additionally, I could create a mappings list of 23 different dictionaries and each dictionary provides a mapping for every unique value in that column to its corresponding numeric value. It is a convenient way to see what a particular numeric label means in the data. Since it is not directly related to the project, I am omitting the code for the mappings list. Figure 3 shows how the data looks like after the encoding is done.



	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	1	5	2	4	1	6	1	0	1	4	...
1	0	5	2	9	1	0	1	0	0	4	...
2	0	0	2	8	1	3	1	0	0	5	...
3	1	5	3	8	1	6	1	0	1	5	...
4	0	5	2	3	0	5	1	1	0	4	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	0	3	2	4	0	5	0	0	0	11	...
8120	0	5	2	4	0	5	0	0	0	11	...
8121	0	2	2	4	0	5	0	0	0	5	...
8122	1	3	3	4	0	8	1	0	1	0	...
8123	0	5	2	4	0	5	0	0	0	11	...

8124 rows × 23 columns

Figure 3

Encoded data

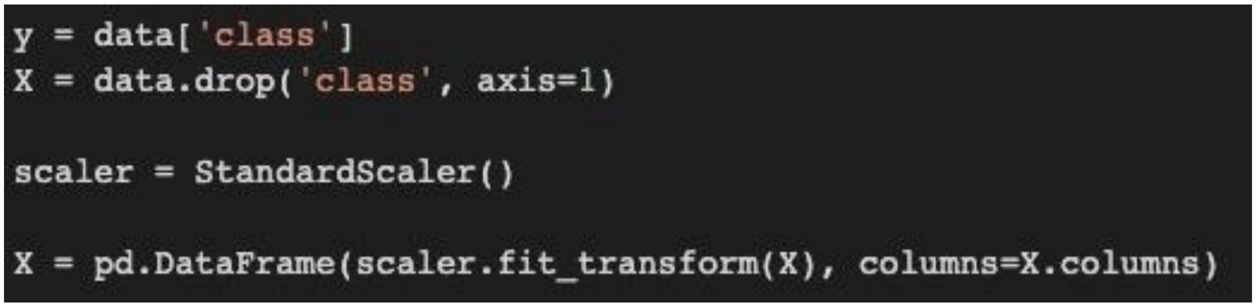
Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022

The next step is to extract the data features into “y” and “X” variables (Figure 4). In machine learning, the "y" variable is often used to represent the target or output variable that we are trying to predict. It is typically the variable that we are trying to model or explain based on the other variables in the dataset. For this data set, the “y” variable is set to the “class” column since it tells us if a mushroom is poisonous (1) or edible (0). The "X" variables, on the other hand, are the input or explanatory variables that are used to predict the value of the target variable. These are the variables that I will use as input to the models, and they are often called "features" or "predictors". In this data set, the “X” variables are the 22 columns, excluding the class column since this is what I am trying to predict.

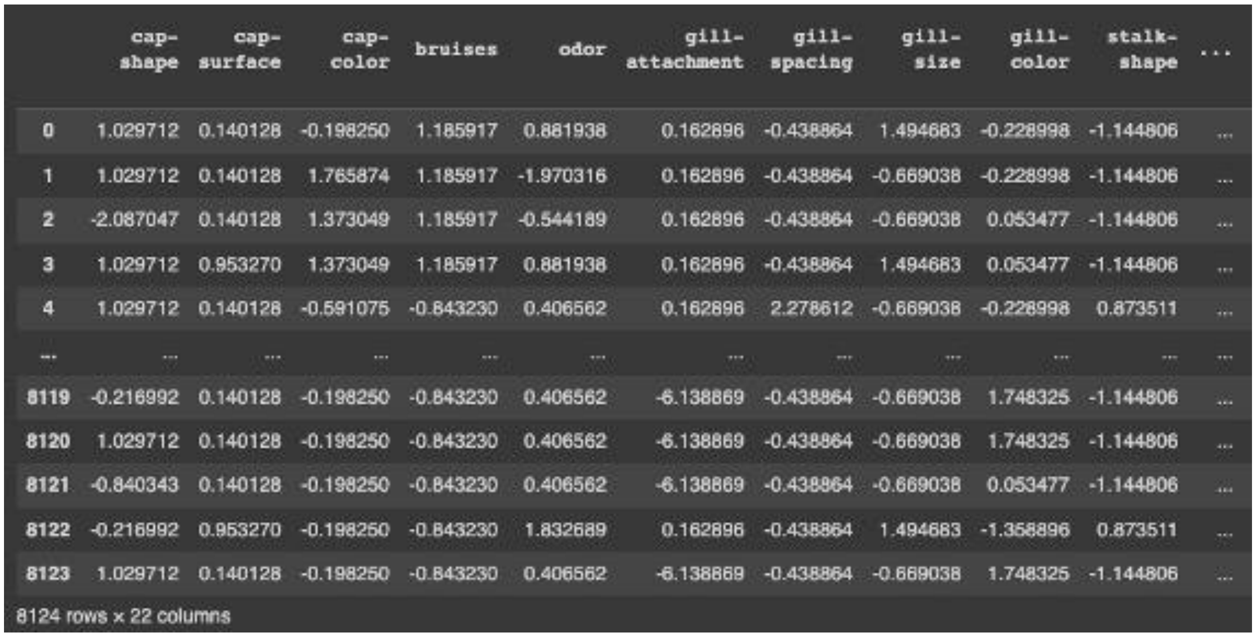


```
y = data['class']
X = data.drop('class', axis=1)

scaler = StandardScaler()

X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

Figure 4

Feature extraction and scaling

The data is not fully ready yet. The next step is to scale these labels in the data. StandardScaler is a tool used in machine learning to transform variables so that they have a mean of 0 and a standard deviation of 1. This is useful because many machine learning algorithms assume that the input data is normally distributed and that the variables have the same scale. For example, if we have a dataset containing measurements of different objects, such as the length and width of a rectangle, the radius of a circle, and the height of a cylinder, we could use StandardScaler to transform all of these variables so that they have the same scale. By scaling the variables in this way, we can help the algorithm to converge faster and produce more accurate results. In Figure 4, an instance of the scaler is fit into the data, and it will then transform X back into the new scaled version. Since the scaler returns a NumPy array, I make it a pandas data frame again to better see the values as in Figure 5. Notice that the new data frame does not contain the class column.



	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...
0	1.029712	0.140128	-0.198250	1.185917	0.881938	0.162896	-0.438864	1.494683	-0.228998	-1.144806	...
1	1.029712	0.140128	1.765874	1.185917	-1.970316	0.162896	-0.438864	-0.669038	-0.228998	-1.144806	...
2	-2.087047	0.140128	1.373049	1.185917	-0.544189	0.162896	-0.438864	-0.669038	0.053477	-1.144806	...
3	1.029712	0.953270	1.373049	1.185917	0.881938	0.162896	-0.438864	1.494683	0.053477	-1.144806	...
4	1.029712	0.140128	-0.591075	-0.843230	0.406562	0.162896	2.278612	-0.669038	-0.228998	0.873511	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	-0.216992	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	1.748325	-1.144806	...
8120	1.029712	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	1.748325	-1.144806	...
8121	-0.840343	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	0.053477	-1.144806	...
8122	-0.216992	0.953270	-0.198250	-0.843230	1.832689	0.162896	-0.438864	1.494683	-1.358896	0.873511	...
8123	1.029712	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	1.748325	-1.144806	...

8124 rows × 22 columns

Figure 5

Scaled data

In machine learning, it is common to split the available data into two sets: a training set and a test set. The training set is used to fit the model, while the test set is used to evaluate the model. The purpose of this split is to evaluate the model's performance on unseen data. If the model is trained and tested on the same data, it is possible that the model may overfit the data, which means that it will perform well on the training data but may not generalize well to new, unseen data. Splitting the data into a training and a test set allows us to evaluate the model's generalization performance and to ensure that the model is not overfitting the data.

After scaling the labels, I can now begin to train-test split the data. The technique works best on large balanced data sets. To verify that the data is balanced, we need to calculate the ratio of positive examples to negative examples in our target "y". By summing up the values in the class column and dividing by its length, 'np.sum(y) / len(y)', the result yields to approximately 0.48. This means that there are 48% of positive examples (edible) and 52% of negative examples (poisonous). Therefore, the data set is balanced, and I can use the train-test procedure. Since my dataset is relatively large, I will choose the train size to be 70% with the remaining of 30% for the test size. We are now ready to select the machine learning models.

Logistic regression is one of the models I am going to use. It is a classification algorithm used to predict a binary outcome. It is used when the dependent variable (the variable we are trying to predict) is binary (i.e., takes only two values such as 0 or 1, poisonous or edible). In logistic regression, the relationship between the independent variables (features) and the dependent variable is modeled using a logistic function. The logistic function takes any input value and maps it to a value between 0 and 1, which can then be interpreted as a probability. For example, if the logistic function outputs a probability of 0.7 for a given input, this can be interpreted as a 70% chance that the dependent variable will take on the value of 1 (i.e., a positive outcome).

Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022

To train a logistic regression model**,** or any other classification model, I need to provide it with a set of labeled training data that was extracted earlier. The model will then learn the relationship between the features and the target by adjusting the model's coefficients. Once trained, the model can be used to make predictions on new, unseen data by applying the learned coefficients to the features of the new data.

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for classification or regression tasks. It works by finding the hyperplane in a high-dimensional space that maximally separates the different classes. For our classification problem, the SVM algorithm finds the hyperplane that maximally separates the different classes (i.e., the one that has the largest distance to the nearest training data points of any class). This hyperplane is called the decision boundary and can be used to classify new data points. The model will then learn the relationship between the features and the target by finding the hyperplane that maximally separates the classes and create predictions on unseen data after being trained.

SVMs are particularly useful when the data is not linearly separable, which means that it cannot be separated into different classes by a straight line. In these cases, the SVM algorithm can use what is called the kernel trick to transform the data into a higher-dimensional space where it becomes linearly separable. There are several different kernel functions that can be used, including the linear kernel, the polynomial kernel, and the radial basis function (RBF) kernel. The RBF kernel is defined as a similarity function that measures the similarity between two points in the input space. It is defined as the exponential of the squared Euclidean distance between the two points, scaled by a constant called the gamma parameter. Since the dataset has a complex, nonlinear relationship between the features and the target, is high-dimensional (many features) and is not linearly separable, the RBF kernel is a great choice in this case.

Lastly, Random Forest is another learning algorithm that can be used for classification or regression tasks. It is an ensemble method, which means that it is based on the combination of multiple weaker models to create a stronger model. In the case of a Random Forest classifier, the model is composed of a number of decision trees. Each decision tree is trained on a random sample of the data, and the predictions made by each tree are combined to make the final prediction. This is done by taking the majority vote of all the trees. For this model, I choose 500 as a number of built decision trees. The higher the number, the better prediction. One of the main advantages of Random Forest is that it is relatively easy to use and provides good performance out of the box. It is also resistant to overfitting, which means that it can generalize well to new data.

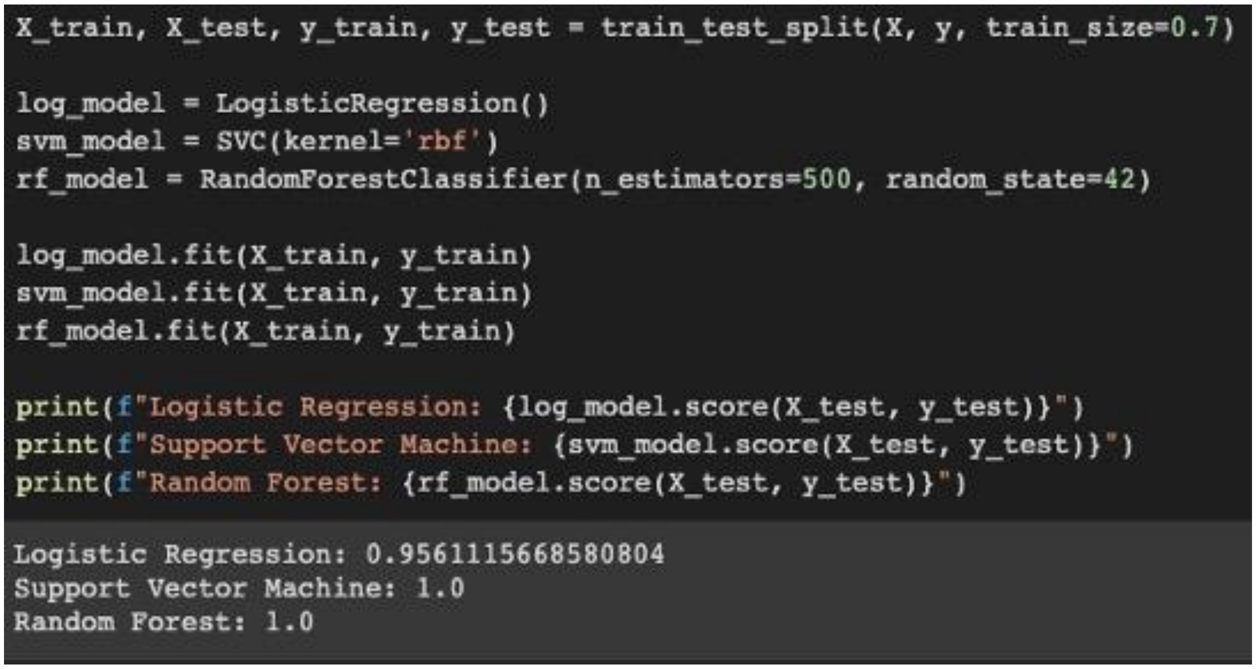
After the models are trained, I can now estimate how well they will perform on unseen data. The ‘score’ method typically takes two arguments: X_test and y_test, which are the features and labels, respectively, of the test set. It then applies the corresponding model to the X_test set and creates predictive values to compare them with the y_test values and to return a score that reflects the model's performance, the accuracy of the model's predictions.

Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022



```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7)

log_model = LogisticRegression()
svm_model = SVC(kernel='rbf')
rf_model = RandomForestClassifier(n_estimators=500, random_state=42)

log_model.fit(X_train, y_train)
svm_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

print(f"Logistic Regression: {log_model.score(X_test, y_test)}")
print(f"Support Vector Machine: {svm_model.score(X_test, y_test)}")
print(f"Random Forest: {rf_model.score(X_test, y_test)}")
```

```
Logistic Regression: 0.9561115668580804
Support Vector Machine: 1.0
Random Forest: 1.0
```

Figure 6

Select and train models to estimate their accuracies

In Figure 6, we see that the accuracies of Support Vector Machine and Random Forest are 100% while Logistic Regression has approximately 96%. To determine why the logistic regression model gives less accuracy than the other two, I had to look back at the labeled training data. Indeed, for the logistic regression to work properly I must add the output label to the “X” variables as well. The output label in a logistic regression model is a binary value that indicates the class of the input data (e.g., 0 or 1). The model uses this labeled training data, which includes both the input features and the output label, to learn the relationship between the input features and the output label, and to make predictions on new, unseen data. After making modifications to the training data, the logistic regression gives a 100% accuracy. To look at how many examples

Yevheniya Solomyana

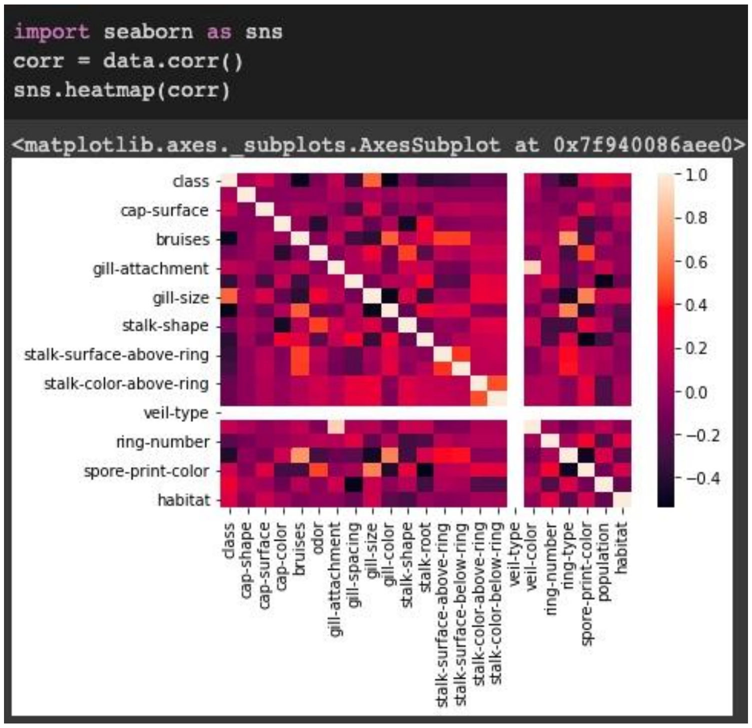
CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022

there where in the X_test set, I run 'X_test.shape' and see that there are 2438 examples. This means that the models got every single example right in predicting the values.

Our last step is to see if any of the features in the dataset dominated over the others in the classification. A correlation matrix is a table that shows the Pearson correlation coefficient between different variables in a dataset. The Pearson correlation coefficient is used to assess the strength and direction of the linear association between two variables. It ranges from -1 to 1, where a value of -1 indicates a strong negative linear relationship, a value of 0 indicates no linear relationship, and a value of 1 indicates a strong positive linear relationship. I am going to use it to identify which variables are strongly correlated with the target variable the most.



```
import seaborn as sns
corr = data.corr()
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f940086aee0>
```

Figure 7

A heatmap of a correlation matrix of the mushroom dataset

Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022

In Figure 7, we see that the heatmap of the correlation matrix does not really give us a concrete feature that has a very strong positive relationship with the target variable class. In general, variables with a Pearson correlation coefficient of 0.7 or higher are considered to be strongly correlated with the target variable, while variables with a Pearson correlation coefficient of 0.3 or lower are considered to be weakly correlated with the target variable. If we look at the heatmap, we see that "gill-size" has a Pearson coefficient of approximately 0.6 with the "class" variable. However, it is not strong enough to say that the gill-size feature has a strong relationship with the target and dominates in the classification models.

If variables in a correlation matrix have no strong correlation with the target variable, it does not necessarily mean that there are no features in the dataset that dominate in the classification accuracy of the models. It is possible for a feature to have a strong relationship with the target variable even if it is not strongly correlated with the other variables in the dataset. In general, the relationship between features and the target variable can be more complex than a simple linear relationship, and a feature that is not strongly correlated with other variables in the dataset can still be important for prediction. For example, a feature that is highly correlated with the target variable in a non-linear fashion (e.g., a quadratic relationship) may be important for prediction even if it is not strongly correlated with other variables in the dataset.

Overall, the three classification models that I have used for this project, Logistic Regression, Support Vector Machine, and Random Forest, yield great accuracies of determining whether a mushroom is poisonous or edible. Moreover, none of the input labels (features) show dominance in predicting at 100% accuracy of the models. In fact, the features are working together as a combination to give these amazing predictions. This means that the mushroom dataset is a very good and clean one and the features are very good predictors of the classes.

Yevheniya Solomyana

CSc 44700 Introduction to Machine Learning

Prof. Erik Grimmelmann

12/19/2022

Citations

Brownlee, Jason. "Train-Test Split for Evaluating Machine Learning Algorithms." *MachineLearningMastery.com*, 26 Aug. 2020, https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

"Choosing a Suitable Machine Learning Algorithm." *GeeksforGeeks*, 17 July 2020, https://www.geeksforgeeks.org/choosing-a-suitable-machine-learning-algorithm/

Kumar, Ajitesh. "Correlation Concepts, Matrix & Heatmap Using Seaborn." *Data Analytics*, 16 Apr. 2022, https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/

"Modelling Probability of Default Using Logistic Regression." *Finance Train*, Finance Train, 23 Sept. 2021, https://financetrain.com/modelling-probability-default-using-logistic-regression

"Pearson Coefficient." *Products Page*, https://agrimetsoft.com/calculators/Pearson%20coefficient

R, Sruthi E. "Random Forest: Introduction to Random Forest Algorithm." *Analytics Vidhya*, 30 Nov. 2022, https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

"Sklearn.preprocessing.LabelEncoder." *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

"Sklearn.preprocessing.StandardScaler." *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

UCI Machine Learning. "Mushroom Classification." *Kaggle*, 1 Dec. 2016, https://www.kaggle.com/datasets/uciml/mushroom-classification