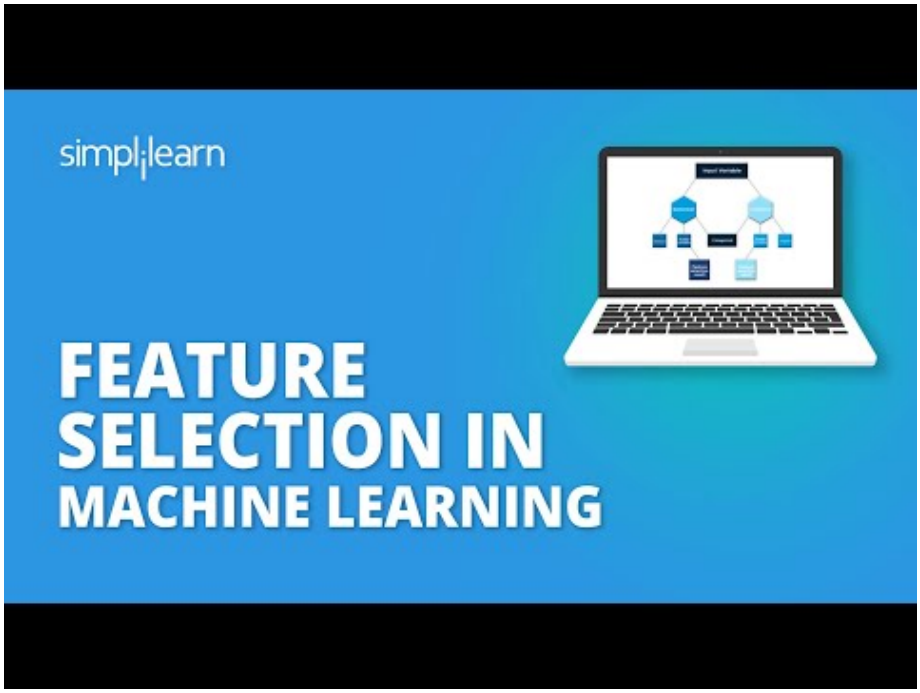
Everything You Need to Know About Feature Selection In Machine Learning

simplilearn.com/tutorials/machine-learning-tutorial/feature-selection-in-machine-learning



The input variables that we give to our machine learning models are called features. Each column in our dataset constitutes a feature. To train an optimal model, we need to make sure that we use only the essential features. If we have too many features, the model can capture the unimportant patterns and learn from noise. The method of choosing the important parameters of our data is called Feature Selection.

In this article titled 'Everything you need to know about Feature Selection', we will teach you all you need to know about feature selection.

Why Feature Selection?

Machine learning models follow a simple rule: whatever goes in, comes out. If we put garbage into our model, we can expect the output to be garbage too. In this case, garbage refers to noise in our data.

To train a model, we collect enormous quantities of data to help the machine learn better. Usually, a good portion of the data collected is noise, while some of the columns of our dataset might not contribute significantly to the performance of our model. Further, having a lot of data can slow down the training process and cause the model to be slower. The model may also learn from this irrelevant data and be inaccurate.

Feature selection is what separates good data scientists from the rest. Given the same model and computational facilities, why do some people win in competitions with faster and more accurate models? The answer is Feature Selection. Apart from choosing the right model for our data, we need to choose the right data to put in our model.

Consider a table which contains information on old cars. The model decides which cars must be crushed for spare parts.

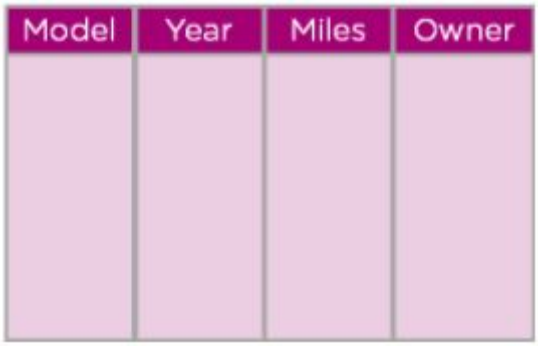


Figure 1: Old cars dataset

In the above table, we can see that the model of the car, the year of manufacture, and the miles it has traveled are pretty important to find out if the car is old enough to be crushed or not. However, the name of the previous owner of the car does not decide if the car should be crushed or not. Further, it can confuse the algorithm into finding patterns between names and the other features. Hence we can drop the column.

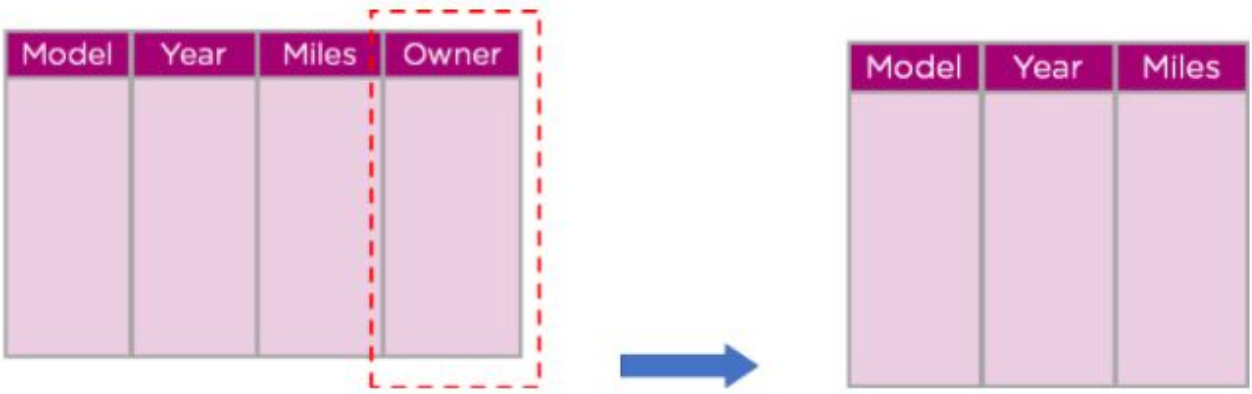


Figure 2: Dropping columns for feature selection

What is Feature Selection?

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data.

It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve. We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data.

Figure 3: Feature Selection

Feature Selection Models

Feature selection models are of two types:

1. Supervised Models: Supervised feature selection refers to the method which uses the output label class for feature selection. They use the target variables to identify the variables which can increase the efficiency of the model
2. Unsupervised Models: Unsupervised feature selection refers to the method which does not need the output label class for feature selection. We use them for unlabelled data.

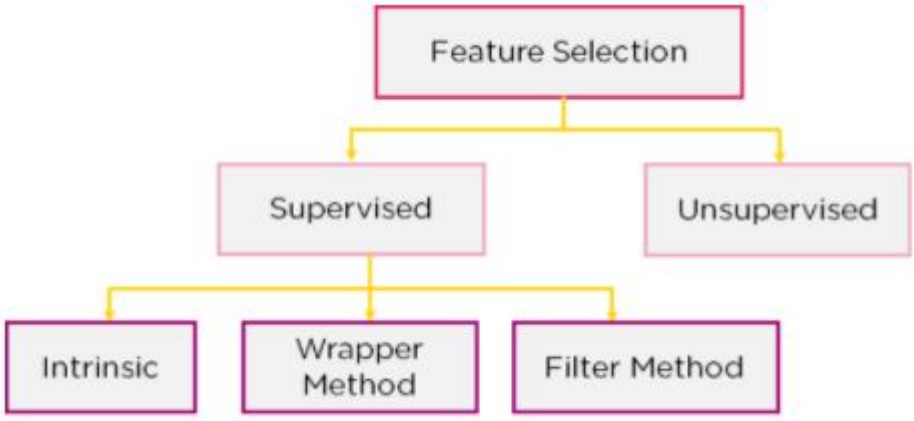


Figure 4: Feature Selection Models

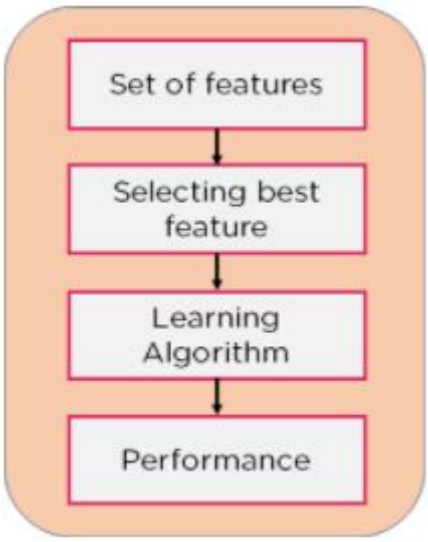
We can further divide the supervised models into three :

1. Filter Method: In this method, features are dropped based on their relation to the output, or how they are **correlating** to the output. We use correlation to check if the features are positively or negatively correlated to the output labels and drop features accordingly. Eg: Information Gain, Chi-Square Test, Fisher's Score, etc.

Figure 5: Filter Method flowchart

2. Wrapper Method: We split our data into subsets and train a model using this. Based on the output of the model, we add and subtract features and train the model again. It forms the subsets using a greedy approach and evaluates the accuracy of all the possible combinations of features. Eg: Forward Selection, Backwards Elimination, etc.

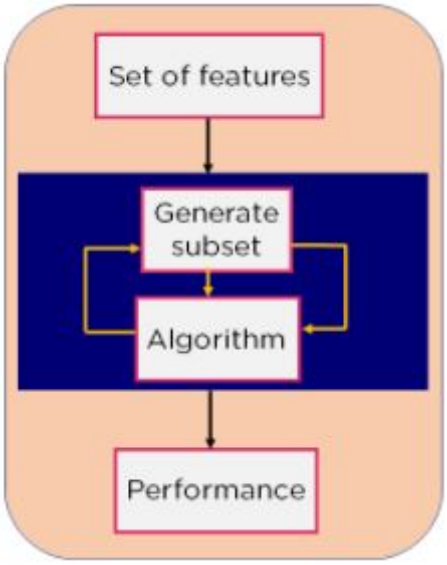
Figure 6: Wrapper Method Flowchart

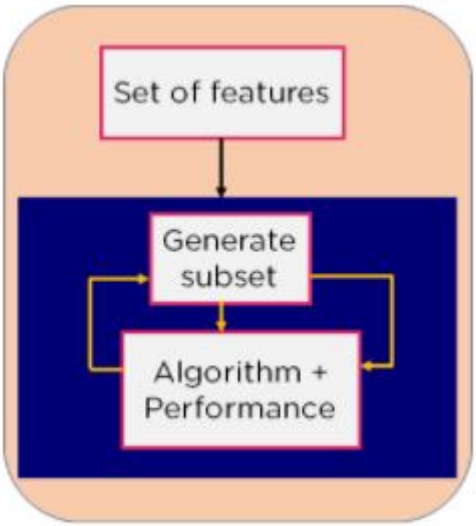


3. Intrinsic Method: This method combines the qualities of both the Filter and Wrapper method to create the best subset.

Figure 7: Intrinsic Model Flowchart

This method takes care of the machine training iterative process while maintaining the computation cost to be minimum. Eg: Lasso and Ridge Regression.





How to Choose a Feature Selection Model?

How do we know which feature selection model will work out for our model? The process is relatively simple, with the model depending on the types of input and output variables.

Variables are of two main types:

- Numerical Variables: Which include integers, float, and numbers.
- Categorical Variables: Which include labels, strings, boolean variables, etc.

Based on whether we have numerical or categorical variables as inputs and outputs, we can choose our feature selection model as follows:

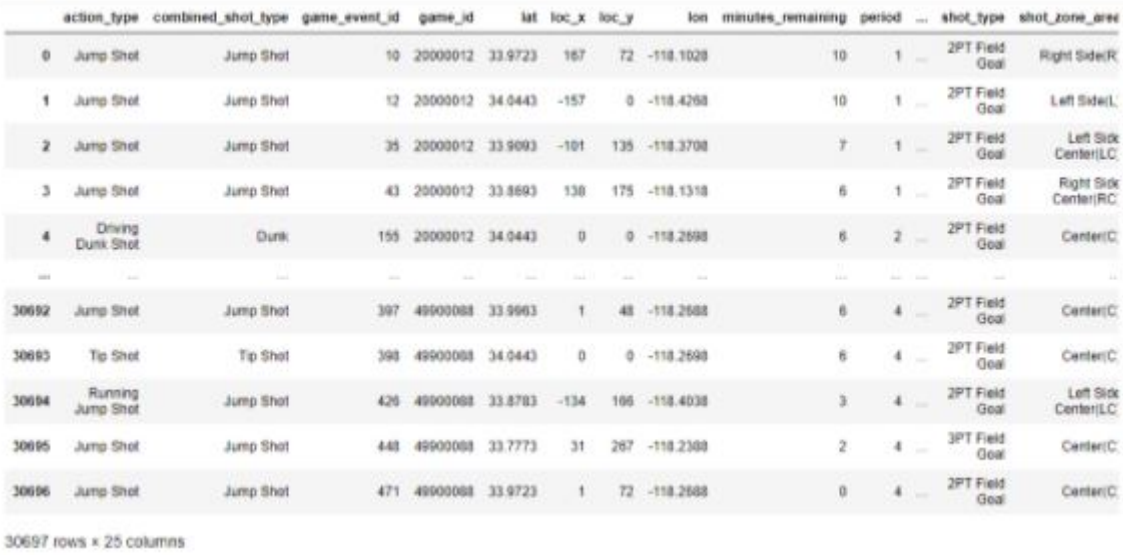
Input Variable	Output Variable	Feature Selection Model
Numerical	Numerical	• Pearson's correlation coefficient • Spearman's rank coefficient
Numerical	Categorical	• ANOVA correlation coefficient (linear). • Kendall's rank coefficient (nonlinear).
Categorical	Numerical	• Kendall's rank coefficient (linear). • ANOVA correlation coefficient (nonlinear).
Categorical	Categorical	• Chi-Squared test (contingency tables). • Mutual Information.

Table 1: Feature Selection Model lookup

Feature Selection With Python

Let's get hands-on experience in feature selection by working on the Kobe Bryant Dataset which analyses shots taken by Kobe from different areas of the court to determine which ones will go into the basket.

The dataset is as shown:



	action_type	combined_shot_type	game_event_id	game_id	lat	loc_x	loc_y	lon	minutes_remaining	period	...	shot_type	shot_zone_area
0	Jump Shot	Jump Shot	10	20000012	33.9723	167	72	-118.1028	10	1	...	2PT Field Goal	Right Side(R
1	Jump Shot	Jump Shot	12	20000012	34.0443	-157	0	-118.4268	10	1	...	2PT Field Goal	Left Side(L
2	Jump Shot	Jump Shot	35	20000012	33.9093	-101	135	-118.3708	7	1	...	2PT Field Goal	Left Side Center(LC
3	Jump Shot	Jump Shot	43	20000012	33.8693	138	175	-118.1318	6	1	...	2PT Field Goal	Right Side Center(RC
4	Driving Dunk Shot	Dunk	155	20000012	34.0443	0	0	-118.2698	6	2	...	2PT Field Goal	Center(C
...	...	...	...	...	...	...	...	...	...	...	...	...	...
30692	Jump Shot	Jump Shot	397	49900088	33.9963	1	48	-118.2688	6	4	...	2PT Field Goal	Center(C
30693	Tip Shot	Tip Shot	398	49900088	34.0443	0	0	-118.2698	6	4	...	2PT Field Goal	Center(C
30694	Running Jump Shot	Jump Shot	426	49900088	33.8783	-134	166	-118.4038	3	4	...	2PT Field Goal	Left Side Center(LC
30695	Jump Shot	Jump Shot	448	49900088	33.7773	31	267	-118.2388	2	4	...	3PT Field Goal	Center(C
30696	Jump Shot	Jump Shot	471	49900088	33.9723	1	72	-118.2688	0	4	...	2PT Field Goal	Center(C

30697 rows × 25 columns

Figure 8: Kobe Bryant Dataset

As we can see, the dataset has 25 different columns. We will not need all of them.

We first begin by loading in the necessary modules.

Figure 9:

Importing modules

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

First, let's check out the loc_x and loc_y columns. They probably represent longitude and latitude.

```
alpha = 0.02
plt.figure(figsize=(10,10))

# loc_x and loc_y
plt.subplot(121)
plt.scatter(data.loc_x, data.loc_y, color='blue', alpha=alpha)
plt.title('loc_x and loc_y')

# lat and lon
plt.subplot(122)
plt.scatter(data.lon, data.lat, color='green', alpha=alpha)
plt.title('lat and lon')
```

Figure 10: Plotting the latitude and longitude columns in our dataset

The figure is as shown:

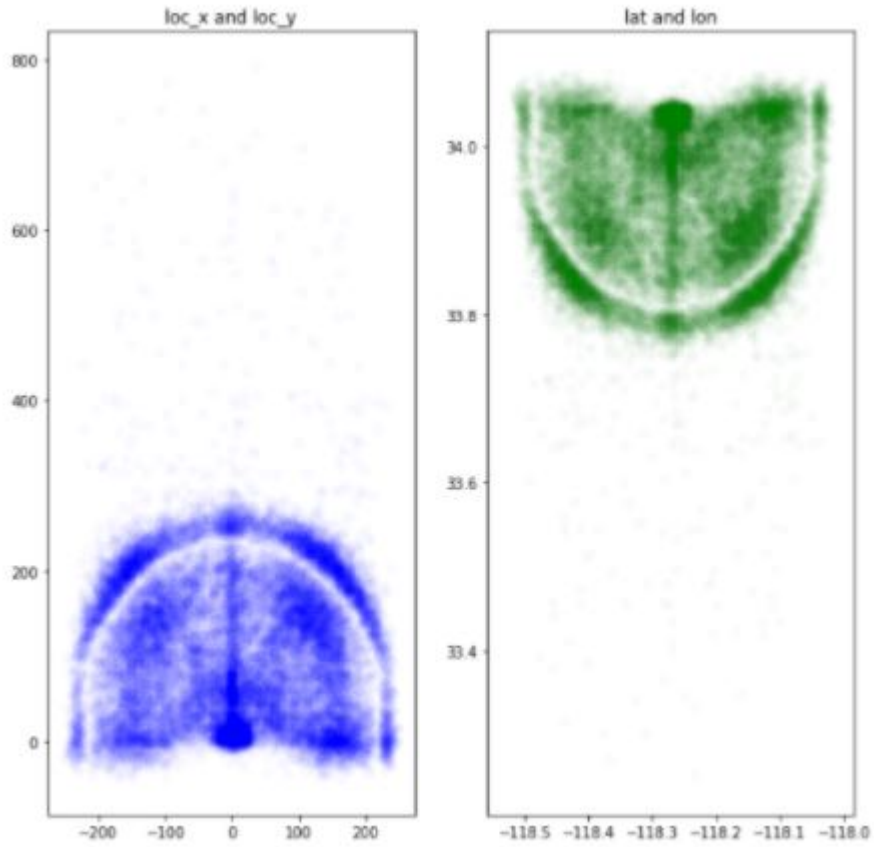


Figure 11: Plotting Latitude and Longitude

From the above figures, we can see that they resemble the two 'D's on a basketball court. Instead of having two separate columns, we can change the coordinates into polar form and have a single column ['angle'].

```
#we get the shape of a basketball court. The top and botton 'D'
#Lets change them into polar cordinates for better analysis
data['dist'] = np.sqrt(data['loc_x']**2 + data['loc_y']**2)

loc_x_zero = data['loc_x'] == 0
data['angle'] = np.array([0]*len(data))
data['angle'][~loc_x_zero] = np.arctan(data['loc_y'][~loc_x_zero] / data['loc_x'][~loc_x_zero])
data['angle'][loc_x_zero] = np.pi / 2
#with these new columns, we don not need the old lat and long columns
```

Figure 12: Changing Latitude and Longitude into polar form

We can combine the minutes and seconds columns into a single column for time.

```
#Lets combine our remaining minutes and seconds columns into one

data['remaining_time'] = data['minutes_remaining'] * 60 + data['seconds_remaining']
data
```

Figure 13: Combining two columns

Let's look at the unique values in the 'team_id' and 'team_name' columns:

Figure 14: Unique values in 'team_id' and 'team_name'

The entire column contains only one value and can be dropped. Let's take a look at the 'match_up' and 'opponent' columns :

```
#Lets Look at team_id and name
print(data['team_id'].unique())
print(data['team_name'].unique())
```

```
[1610612747]
['Los Angeles Lakers']
```

```
#Now let's look at matchup and opponent
pd.DataFrame({'matchup':data.matchup, 'opponent':data.opponent})
```

	matchup	opponent
0	LAL @ POR	POR
1	LAL @ POR	POR
2	LAL @ POR	POR
3	LAL @ POR	POR
4	LAL @ POR	POR
...	...	...
30692	LAL vs. IND	IND
30693	LAL vs. IND	IND
30694	LAL vs. IND	IND
30695	LAL vs. IND	IND
30696	LAL vs. IND	IND

30697 rows × 2 columns

Figure 15: 'match_up' and 'opponent' columns

Again, they contain the same information. Let's plot the values of 'dist' and 'shot_distance' columns on the same graph to see how they differ:

```
#The basically contain the same info
#Lts us see how distance and shot_distance differ

plt.figure(figsize=(5,5))

plt.scatter(data.dist, data.shot_distance, color='blue')
plt.title('dist and shot_distance')
```

```
Text(0.5, 1.0, 'dist and shot_distance')
```

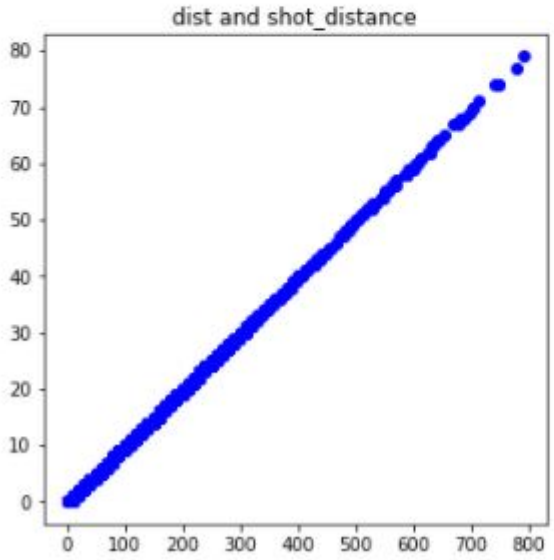


Figure 16: Plotting ‘dist’ and ‘shot_distance’ columns

Again, they contain exactly the same information. Let’s take a look at columns shot_zone_area, shot_zone_basic and shot_zone_range.

```
#again they contain the same info
#Lte's look at shot_zone_area, shot_zone_basic, shot_zone_range

import matplotlib.cm as cm
plt.figure(figsize=(20,10))

def scatter_plot_by_category(feat):
    alpha = 0.1
    gs = data.groupby(feat)
    cs = cm.rainbow(np.linspace(0, 1, len(gs)))
    for g, c in zip(gs, cs):
        plt.scatter(g[1].loc_x, g[1].loc_y, color=c, alpha=alpha)

# shot_zone_area
plt.subplot(131)
scatter_plot_by_category('shot_zone_area')
plt.title('shot_zone_area')

# shot_zone_basic
plt.subplot(132)
scatter_plot_by_category('shot_zone_basic')
plt.title('shot_zone_basic')

# shot_zone_range
plt.subplot(133)
scatter_plot_by_category('shot_zone_range')
plt.title('shot_zone_range')
```

Figure 17: Plotting the different shot zones columns

The figure depicted below shows the plots :

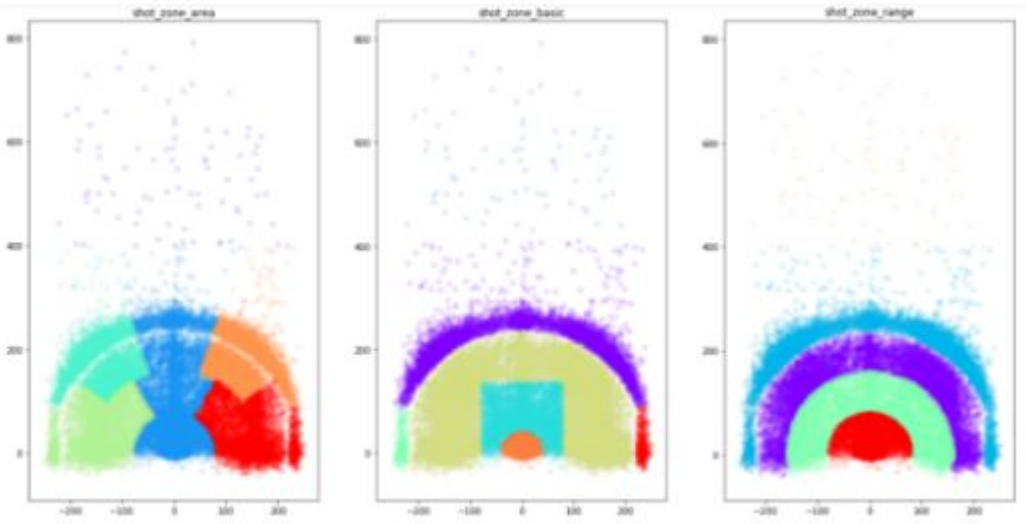


Figure 18: Different shot zones

We can see that they contain the different parts of the court from where the shots were taken. This information already exists in the angle and dist columns.

Now, let’s drop all the useless columns.

```
#They are regions of the court, But we have already stored this info in angle and dist columns

#now let's drop the useless columns
drops = ['shot_id', 'team_id', 'team_name', 'shot_zone_area', 'shot_zone_range', 'shot_zone_basic', \
         'matchup', 'lon', 'lat', 'seconds_remaining', 'minutes_remaining', \
         'shot_distance', 'loc_x', 'loc_y', 'game_event_id', 'game_id', 'game_date']
for drop in drops:
    data = data.drop(drop, 1)
```

Figure 19: Dropping Columns

After merging columns and removing useless columns, we get a dataset that contains only 11 important columns.

	action_type	combined_shot_type	period	playoffs	season	shot_made_flag	shot_type	opponent	dist	angle	remaining_time
0	Jump Shot	Jump Shot	1	0	2000-01	NaN	2PT Field Goal	POR	181.859836	0.407058	627
1	Jump Shot	Jump Shot	1	0	2000-01	0.0	2PT Field Goal	POR	157.000000	-0.000000	622
2	Jump Shot	Jump Shot	1	0	2000-01	1.0	2PT Field Goal	POR	168.920119	-0.928481	465
3	Jump Shot	Jump Shot	1	0	2000-01	0.0	2PT Field Goal	POR	222.865430	0.903063	412
4	Driving Dunk Shot	Dunk	2	0	2000-01	1.0	2PT Field Goal	POR	0.000000	1.570796	379
...	...	...	...	...	...	...	...	...	...	...	...
30692	Jump Shot	Jump Shot	4	1	1999-00	0.0	2PT Field Goal	IND	48.010416	1.549966	365
30693	Tip Shot	Tip Shot	4	1	1999-00	NaN	2PT Field Goal	IND	0.000000	1.570796	365
30694	Running Jump Shot	Jump Shot	4	1	1999-00	1.0	2PT Field Goal	IND	213.335417	-0.891663	208
30695	Jump Shot	Jump Shot	4	1	1999-00	0.0	3PT Field Goal	IND	268.793601	1.455209	130
30696	Jump Shot	Jump Shot	4	1	1999-00	0.0	2PT Field Goal	IND	72.006944	1.556908	39

30697 rows × 11 columns

Figure 20: Final Dataset

> Learn the essentials of object-oriented programming, web development with Django, and more with the Python Training Course. Enroll now!

Conclusion

In this article titled ‘Everything you need to know about Feature Selection’, we got an idea of how important it is to select the best features for our machine learning model. We then took a look at what feature selection is and some feature selection models. We then moved onto a simple way to choose the right feature selection model based on the input and output values. Finally, we saw how to implement feature selection in Python with a demo. If you are looking to learn more about feature selection and related fundamental features of Python, Simplielarn’s Python Certification Course would be ideal for you. This python certification course covers the basics fundamentals of python including data operations, conditional statements, shell scripting, and Django and much more, and prepares you for a rewarding career as a professional Python programmer.

Was this article on feature selection useful to you? Do you have any doubts or questions for us? Mention them in this article's comments section, and we'll have our experts answer them for you at the earliest!