Machine Learning with PyTorch and Scikit-Learn

– Code Examples

Package version checks

Add folder to path in order to load from the check_packages.py script:

```
import sys
sys.path.insert(0, '..')
```

Check recommended package versions:

```
from python_environment_check import check_packages


d = {
    'numpy': '1.21.2',
    'matplotlib': '3.4.3',
    'sklearn': '1.0',
    'pandas': '1.3.2'
}
check_packages(d)
```

```
[OK] numpy 1.21.2
[OK] matplotlib 3.4.3
[OK] sklearn 1.0.2
[OK] pandas 1.3.2
```

Chapter 4 - Building Good Training Datasets – Data Preprocessing

Overview

```
from IPython.display import Image
%matplotlib inline
```

Dealing with missing data

Identifying missing values in tabular data

This code is using the Pandas library to read a CSV data file from a string. The string is defined as `csv_data`, which contains a 4-column data table.

The code then checks the version of Python being used and converts the string to unicode if the version is 2.7 or earlier.

Finally, the code creates a Pandas DataFrame `df` by using the `pd.read_csv` function and passing in the string data using the `StringIO` function from the `io` library. The resulting DataFrame `df` contains the data from the CSV string.

```
import pandas as pd
from io import StringIO
import sys

csv_data = \
'''A,B,C,D
1.0,2.0,3.0,4.0
```

```
5.0,6.0,,8.0
10.0,11.0,12.0,'''

# If you are using Python 2.7, you need
# to convert the string to unicode:

if (sys.version_info < (3, 0)):
    csv_data = unicode(csv_data)

df = pd.read_csv(StringIO(csv_data))
df
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

```
# import pandas as pd
# from io import StringIO
# import sys

csv_data = \
'''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,
13.0,14.0,,'''

# If you are using Python 2.7, you need
# to convert the string to unicode:

# if (sys.version_info < (3, 0)):
#     csv_data = unicode(csv_data)

df2 = pd.read_csv(StringIO(csv_data))
df2
```

	A	B	C	D
0	1.0	2.0	3.0	4.0

	A	B	C	D
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN
3	13.0	14.0	NaN	NaN

The second code block uses the `isnull()` method from the pandas library to check for missing values in the dataframe `df`. The `sum()` method is then applied on the resulting boolean dataframe to count the number of missing values in each column. The result is a pandas series that displays the count of missing values in each column.

```
df.isnull()
```

	A	B	C	D
0	False	False	False	False
1	False	False	True	False
2	False	False	False	True

```
df.isnull().sum()
```

```
A    0
B    0
C    1
D    1
dtype: int64
```

The code `df.values` accesses the underlying NumPy array of the Pandas DataFrame `df`. The `values` attribute returns a NumPy array representation of the DataFrame, allowing for easy integration with other numerical libraries in Python. This can be useful for performing numerical operations or passing the data to a machine learning model.

```
# access the underlying NumPy array
# via the `values` attribute
df.values
```

```
array([[ 1.,  2.,  3.,  4.],
       [ 5.,  6., nan,  8.],
       [10., 11., 12., nan]])
```

Eliminating training examples or features with missing values

```
csv_data = \
'''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,
13.0,14.0,,'''

df2 = pd.read_csv(StringIO(csv_data))
df2
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN
3	13.0	14.0	NaN	NaN

```
df
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

The code is removing rows that contain missing values from the dataframe "df". The "dropna" method is used to drop missing values in the dataframe. The "axis" parameter is set to 0, which indicates that we want to drop the rows that contain missing values. The resulting dataframe will only contain rows with no missing values.

```
# remove rows that contain missing values

df.dropna(axis=0)
```

	A	B	C	D
0	1.0	2.0	3.0	4.0

```
# remove rows that contain missing values

df2.dropna(axis=0)
```

	A	B	C	D
0	1.0	2.0	3.0	4.0

```
# remove columns that contain missing values

df.dropna(axis=1)
```

	A	B
0	1.0	2.0
1	5.0	6.0
2	10.0	11.0

```
# remove columns that contain missing values

df2.dropna(axis=1)
```

	A	B
0	1.0	2.0
1	5.0	6.0
2	10.0	11.0
3	13.0	14.0

```
df
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

The code `df.dropna(how='all')` will drop only the rows where all columns contain NaN values. In this case, if a row has at least one non-NaN value, it will be kept and not dropped.

This is useful when you want to remove only the rows that have no information at all, and keep the ones that contain at least some data.

```
# only drop rows where all columns are NaN

df.dropna(how='all')
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

The code `df.dropna(thresh=4)` will drop rows in the pandas DataFrame `df` that have less than 4 non-null values. The argument thresh specifies the minimum number of non-null values a row must have in order to be kept. In this case, `thresh=4` means that only rows with 4 or more non-null values will be kept, and any row with less than 4 non-null values will be dropped.

```
# drop rows that have fewer than 4 real values

df.dropna(thresh=4)
```

	A	B	C	D
0	1.0	2.0	3.0	4.0

```
# only drop rows where NaN appear in specific columns (here: 'C')

df.dropna(subset=['C'])
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
2	10.0	11.0	12.0	NaN

Imputing missing values

```
# again: our original array
df.values
```

```
array([[ 1.,  2.,  3.,  4.],
       [ 5.,  6., nan,  8.],
       [10., 11., 12., nan]])
```

```
df
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

Impute Missing Values via the Column Mean

The code below uses the SimpleImputer class from scikit-learn's `impute` module to fill in missing values in a pandas DataFrame `df` with the mean value of each column. The missing values are represented by `np.nan`.

```
from sklearn.impute import SimpleImputer
import numpy as np

# Create an instance of the SimpleImputer class
imr = SimpleImputer(missing_values=np.nan, strategy='mean')

# Fit the imputer to the data
imr = imr.fit(df.values)

# Transform the data to fill in the missing values
imputed_data = imr.transform(df.values)

# Preview the imputed data
print(imputed_data)

::: {.cell execution_count=274}
```

```
```{.python .cell-code}
df
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

:::

```
# impute missing values via the column mean

from sklearn.impute import SimpleImputer
import numpy as np

imr = SimpleImputer(missing_values=np.nan, strategy='mean')
imr = imr.fit(df.values)
imputed_data = imr.transform(df.values)
imputed_data
```

```
array([[ 1. ,  2. ,  3. ,  4. ],
       [ 5. ,  6. ,  7.5,  8. ],
       [10. , 11. , 12. ,  6. ]])
```

```
print(imr.statistics_)
```

```
[5.33333333 6.33333333 7.5        6.        ]
```

```
df
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN
```

The code `df.fillna(df.mean())` replaces the missing values in the pandas dataframe `df` with the mean value of each column.

This code first computes the mean of each column by calling `df.mean()`. The resulting Series is then passed to the fillna method, which replaces the missing values in df with the corresponding mean values.

```
df.mean()
```

```
A    5.333333
B    6.333333
C    7.500000
D    6.000000
dtype: float64
```

```
df.fillna(df.mean())
```

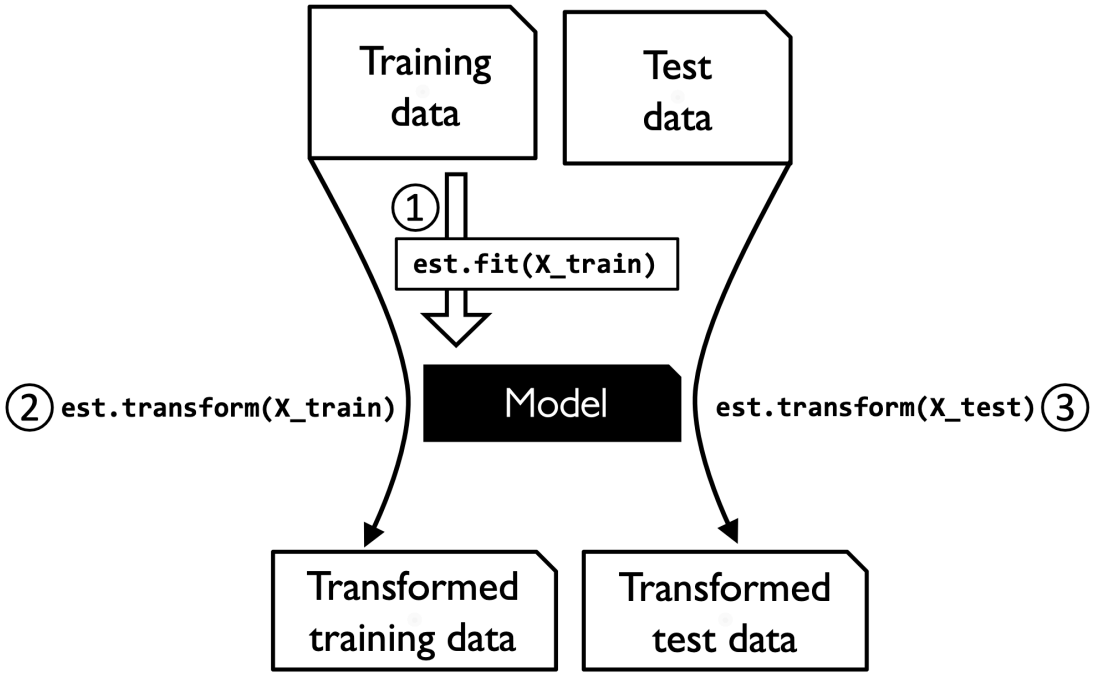
	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.5	8.0
2	10.0	11.0	12.0	6.0

```
df
```

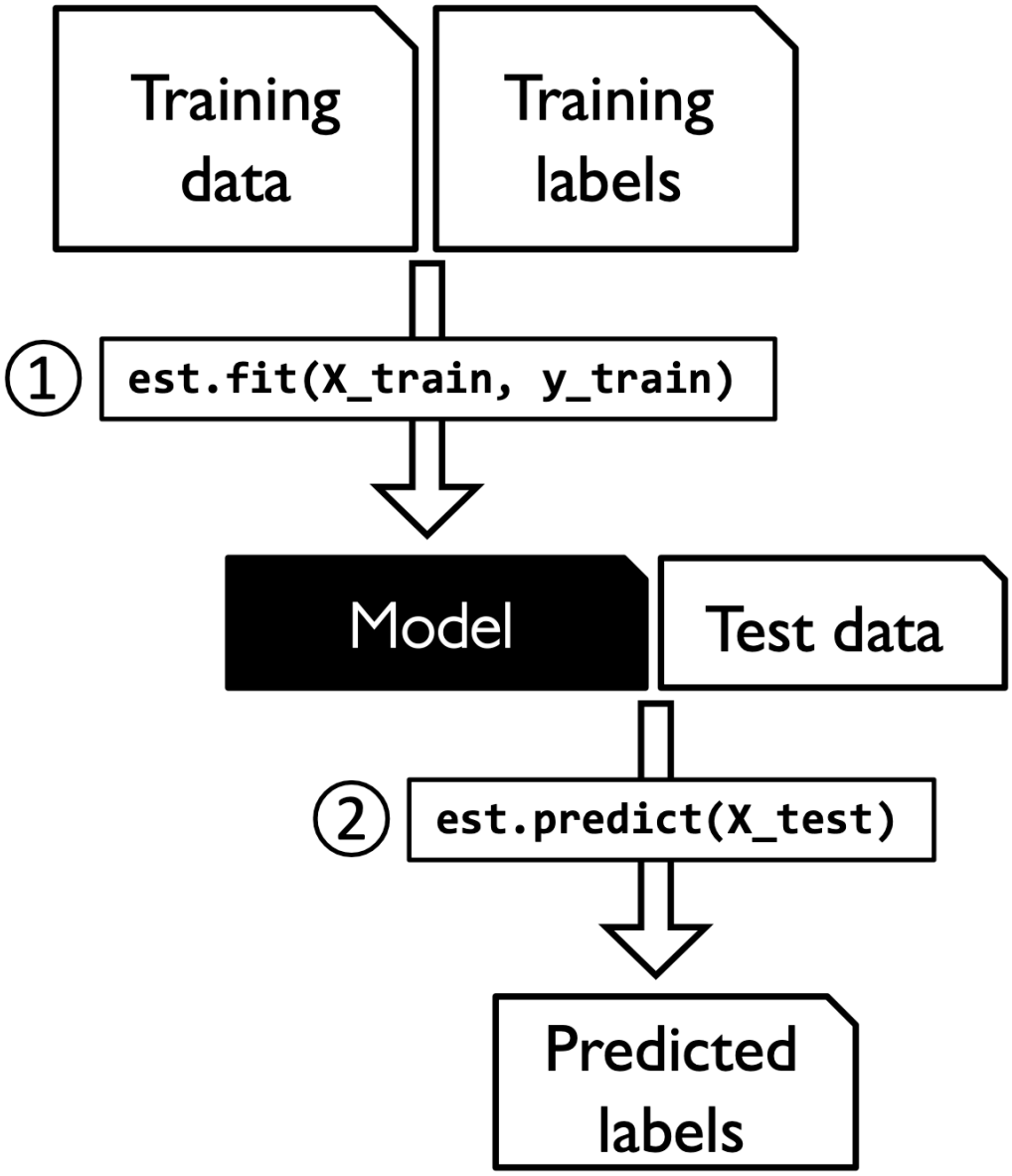
	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

Understanding the scikit-learn estimator API

```
Image(filename='figures/04_02.png', width=400)
```




```
Image(filename='figures/04_03.png', width=300)
```


Training data
Training labels
① est.fit(X_train, y_train)
Model
Test data
② est.predict(X_test)
Predicted labels

Handling categorical data

Nominal and ordinal features

This code creates a Pandas dataframe from a list of lists, where each list represents a row in the dataframe. The dataframe contains information about a set of items, with columns for 'color', 'size', 'price', and 'classlabel'. The columns are given names using the `.columns` attribute of the dataframe. The resulting dataframe is then stored in the variable `df`.

```
import pandas as pd

df = pd.DataFrame([['green', 'M', 10.1, 'class2'],
                   ['red', 'L', 13.5, 'class1'],
                   ['blue', 'XL', 15.3, 'class2']])

df.columns = ['color', 'size', 'price', 'classlabel']
df
```

	color	size	price	classlabel
0	green	M	10.1	class2
1	red	L	13.5	class1
2	blue	XL	15.3	class2

Mapping ordinal features

```
df
```

	color	size	price	classlabel
0	green	M	10.1	class2
1	red	L	13.5	class1
2	blue	XL	15.3	class2

This code creates a dictionary called size_mapping that maps the string values 'XL', 'L', and 'M' to integers 3, 2, and 1, respectively. The code then uses the map method from the Pandas DataFrame df to replace the values in the 'size' column with the corresponding integers in the size_mapping dictionary.

For example, the first row of the 'size' column has the value 'M', so it will be replaced with 1. The second row has the value 'L', so it will be replaced with 2, and so on. The modified DataFrame df is then displayed at the end of the code.

```
size_mapping = {'XL': 3,
                'L': 2,
                'M': 1}

df['size'] = df['size'].map(size_mapping)
df
```

	color	size	price	classlabel
0	green	1	10.1	class2
1	red	2	13.5	class1
2	blue	3	15.3	class2

The line `df['size'].map(inv_size_mapping)` maps the numerical values back to the original categorical values using the 'inv_size_mapping' dictionary, which is created using the line `inv_size_mapping = {v: k for k, v in size_mapping.items()}` by swapping the keys and values of the 'size_mapping' dictionary.

```
df
```

	color	size	price	classlabel
0	green	1	10.1	class2
1	red	2	13.5	class1
2	blue	3	15.3	class2

```
size_mapping.items()
```

```
dict_items([('XL', 3), ('L', 2), ('M', 1)])
```

```
inv_size_mapping = {v: k for k, v in size_mapping.items()}
print(inv_size_mapping)
```

```
{3: 'XL', 2: 'L', 1: 'M'}
```

```
inv_size_mapping = {v: k for k, v in size_mapping.items()}
df['size'].map(inv_size_mapping)
```

```
0     M
1     L
2    XL
Name: size, dtype: object
```

```
df
```

	color	size	price	classlabel
0	green	1	10.1	class2
1	red	2	13.5	class1
2	blue	3	15.3	class2

Encoding class labels

This code is creating a dictionary called `class_mapping` that maps the unique class labels in the classlabel column of the DataFrame `df` to integers. The `np.unique` function is used to extract the unique class labels from the classlabel column. The `enumerate` function is used to generate a sequence of pairs, where the first element of each pair is an index, and the second element of each pair is a class label. The resulting dictionary maps each class label to a unique integer.

```
import numpy as np

# create a mapping dict
# to convert class labels from strings to integers
class_mapping = {label: idx for idx, label in enumerate(np.unique(df['classlabel']))}
class_mapping
```

```
{'class1': 0, 'class2': 1}
```

```
df
```

	color	size	price	classlabel
0	green	1	10.1	class2
1	red	2	13.5	class1
2	blue	3	15.3	class2

This code uses the `map` method of a Pandas DataFrame column, which applies a given mapping to each element in the column. In this case, the `classlabel` column is being transformed from strings to integers, using the `class_mapping` dictionary that was created earlier. The `class_mapping` dictionary maps each unique string class label in the classlabel column to an integer index, which is stored as the corresponding value in the dictionary.

```
# to convert class labels from strings to integers
df['classlabel'] = df['classlabel'].map(class_mapping)
df
```

	color	size	price	classlabel
0	green	1	10.1	1
1	red	2	13.5	0
2	blue	3	15.3	1

```
# reverse the class label mapping
inv_class_mapping = {v: k for k, v in class_mapping.items()}
df['classlabel'] = df['classlabel'].map(inv_class_mapping)
df
```

	color	size	price	classlabel
0	green	1	10.1	class2
1	red	2	13.5	class1
2	blue	3	15.3	class2

The code below is using the `LabelEncoder` class from the `sklearn.preprocessing` module to perform label encoding on the class labels stored in the column `'classlabel'` of the pandas DataFrame `df`.

The `fit_transform` method of the `LabelEncoder` class is used to fit the label encoder to the class labels and then perform the transformation to convert the class labels from strings to integers. The transformed integer representation of the class labels is stored in the variable y.

```
from sklearn.preprocessing import LabelEncoder

# Label encoding with sklearn's LabelEncoder
class_le = LabelEncoder()
y = class_le.fit_transform(df['classlabel'].values)
y
```

```
array([1, 0, 1])
```

```
# reverse mapping
class_le.inverse_transform(y)
```

```
array(['class2', 'class1', 'class2'], dtype=object)
```

Performing one-hot encoding on nominal features

```
df
```

	color	size	price	classlabel
0	green	1	10.1	class2
1	red	2	13.5	class1
2	blue	3	15.3	class2

This code is preprocessing the data by converting the categorical feature "color" into numerical values. The `LabelEncoder` class from the `sklearn.preprocessing` library is used for this purpose.

First, the feature data is extracted from the Pandas dataframe `df` into a NumPy array `X`. The first column of `X`, which corresponds to the "color" feature, is then transformed using the `fit_transform` method of the `LabelEncoder` class. This method fits the encoder to the categorical feature data and returns a transformed array of numerical values.

After the transformation, the first column of `X` now contains numerical values that represent the categorical feature "color".

```
X = df[['color', 'size', 'price']].values
color_le = LabelEncoder()
```

```
X[:, 0] = color_le.fit_transform(X[:, 0])
X
```

```
array([[1, 1, 10.1],
       [2, 2, 13.5],
       [0, 3, 15.3]], dtype=object)
```

This code performs one-hot encoding on the first column of the feature matrix `X`(which is the 'color' column) using the `OneHotEncoder` class from scikit-learn. The purpose of one-hot encoding is to transform categorical features, which are represented as integers, into a binary matrix representation.

The fit_transform method of the OneHotEncoder class takes in a 2D array as input, so we reshape the 1D array `X[:, 0]` into a 2D array `X[:, 0].reshape(-1, 1)` to make sure it fits the input format of the `fit_transform` method. The `toarray` method is used to convert the sparse matrix returned by the fit_transform method into a dense NumPy array.

```
X = df[['color', 'size', 'price']].values
print(X)
```

```
[['green' 1 10.1]
 ['red' 2 13.5]
 ['blue' 3 15.3]]
```

```
from sklearn.preprocessing import OneHotEncoder

X = df[['color', 'size', 'price']].values
color_ohe = OneHotEncoder()
color_ohe.fit_transform(X[:, 0].reshape(-1, 1)).toarray()
```

```
array([[0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.]])
```

```
df
```

	color	size	price	classlabel
0	green	1	10.1	class2
1	red	2	13.5	class1
2	blue	3	15.3	class2

This code uses `ColumnTransformer` from scikit-learn to apply different preprocessing techniques to different columns in the input feature array `X`. The `ColumnTransformer` takes a list of transformers as argument, where each transformer is defined by a name, an instance of a transformer class, and a list of indices specifying which columns to apply the transformer to. In this case, the first transformer is an instance of `OneHotEncoder` and is applied to the first column of `X` (the color feature). The second transformer is specified as `'nothing'` with `'passthrough'` as the class and is applied to the second and third columns of X (the size and price features, respectively). The `ColumnTransformer` applies the specified transformers to the columns in `X` and returns the transformed array. The resulting array is then cast to float type using `astype(float)`.

```
from sklearn.compose import ColumnTransformer

X = df[['color', 'size', 'price']].values
c_transf = ColumnTransformer([ ('onehot', OneHotEncoder(), [0]),
                               ('nothing', 'passthrough', [1, 2])])
c_transf.fit_transform(X).astype(float)
```

```
array([[ 0. ,  1. ,  0. ,  1. , 10.1],
       [ 0. ,  0. ,  1. ,  2. , 13.5],
       [ 1. ,  0. ,  0. ,  3. , 15.3]])
```

The code is creating a one-hot encoding of the data in the `df` dataframe for the columns `'price'`, `'color'`, and `'size'` using pandas' `get_dummies` function. One-hot encoding is a process of converting categorical variables into a numerical representation, where each category value is represented as a separate binary column (also known as a "dummy variable"). The new columns created by the get_dummies function are binary columns, each representing one category of the original categorical column. The get_dummies function returns the transformed data as a new dataframe, which can be used for machine learning tasks.

```
# one-hot encoding via pandas

pd.get_dummies(df[['price', 'color', 'size']])
```

	price	size	color_blue	color_green	color_red
0	10.1	1	0	1	0
1	13.5	2	0	0	1
2	15.3	3	1	0	0

The `drop_first` parameter in `pd.get_dummies` is used to prevent multicollinearity. Multicollinearity is a phenomenon in which two or more predictors in a regression model are highly correlated, causing the coefficients to become unstable. In the one-hot encoding process, the first category of a categorical variable is dropped to avoid the dummy variable trap, which is another way to prevent multicollinearity. By setting drop_first=True, the first category of each categorical variable is dropped from the encoding process, reducing the number of columns in the resulting dataframe. This can also help to reduce the risk of overfitting, as it reduces the number of predictors in the model.

```
# multicollinearity guard in get_dummies

pd.get_dummies(df[['price', 'color', 'size']], drop_first=True)
```

	price	size	color_green	color_red
0	10.1	1	1	0
1	13.5	2	0	1
2	15.3	3	0	0

This code is using the OneHotEncoder method from the scikit-learn library to perform one-hot encoding on the "color" feature in the input data `X`. The input data `X` is a matrix or 2D array-like object with three columns (features), and the `OneHotEncoder` will transform the first column (index 0) into multiple columns, one for each unique category in that column.

The code also uses the `ColumnTransformer` class from scikit-learn, which allows you to apply different transformations to different columns of your input data. In this case, the `ColumnTransformer` is being used to apply the one-hot encoding transformation only to the "color" feature (column 0), while leaving the other two features unchanged.

The `OneHotEncoder` is instantiated with two parameters: "categories='auto' " means that the categories to encode should be automatically inferred from the input data, and "drop='first' " means that the first category in each feature should be dropped to reduce multicollinearity.

Finally, the `fit_transform` method of the ColumnTransformer is used to fit the one-hot encoder to the input data and then transform the data. The resulting transformed data is then cast to type float to ensure that all values are numerical.

```
# multicollinearity guard for the OneHotEncoder

color_ohe = OneHotEncoder(categories='auto', drop='first')
c_transf = ColumnTransformer([ ('onehot', color_ohe, [0]),
                               ('nothing', 'passthrough', [1, 2])])
c_transf.fit_transform(X).astype(float)
```

```
array([[ 1. ,  0. ,  1. , 10.1],
       [ 0. ,  1. ,  2. , 13.5],
       [ 0. ,  0. ,  3. , 15.3]])
```

Optional: Encoding Ordinal Features

If we are unsure about the numerical differences between the categories of ordinal features, or the difference between two ordinal values is not defined, we can also encode them using a threshold encoding with 0/1 values. For example, we can split the feature "size" with values M, L, and XL into two new features "x > M" and "x > L". Let's consider the original DataFrame:

```
df = pd.DataFrame([['green', 'M', 10.1, 'class2'],
                   ['red', 'L', 13.5, 'class1'],
                   ['blue', 'XL', 15.3, 'class2']])

df.columns = ['color', 'size', 'price', 'classlabel']
df
```

	color	size	price	classlabel
0	green	M	10.1	class2
1	red	L	13.5	class1
2	blue	XL	15.3	class2

We can use the `apply` method of pandas' DataFrames to write custom lambda expressions in order to encode these variables using the value-threshold approach:

```
df['x > M'] = df['size'].apply(lambda x: 1 if x in {'L', 'XL'} else 0)
df['x > L'] = df['size'].apply(lambda x: 1 if x == 'XL' else 0)

del df['size']
df
```

	color	price	classlabel	x > M	x > L
0	green	10.1	class2	0	0
1	red	13.5	class1	1	0
2	blue	15.3	class2	1	1

Partitioning a dataset into a separate training and test set

```
df_wine = pd.read_csv('https://archive.ics.uci.edu/'
                      'ml/machine-learning-databases/wine/wine.data',
                      header=None)

# if the Wine dataset is temporarily unavailable from the
# UCI machine learning repository, un-comment the following line
# of code to load the dataset from a local path:

# df_wine = pd.read_csv('wine.data', header=None)


df_wine.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
                   'Alcalinity of ash', 'Magnesium', 'Total phenols',
                   'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
                   'Color intensity', 'Hue', 'OD280/OD315 of diluted wines',
                   'Proline']

print('Class labels', np.unique(df_wine['Class label']))
df_wine.head()
```

```
Class labels [1 2 3]
```

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	No
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.2
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.2
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.3
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.2
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.3

This code is a demonstration of how to split a dataset into training and testing sets using the `train_test_split` function from the `sklearn.model_selection` module in scikit-learn.

The first line extract the feature values and target values from the `df_wine` dataframe into `X` and `y` respectively.

In the next line, the `train_test_split` function is used to split `X` and `y` into training and testing sets. The function takes several parameters:

- `X` and `y`: the feature values and target values that we want to split.
- `test_size`: the proportion of the data that should be allocated to the test set. In this case, 0.3 means that 30% of the data will be allocated to the test set and 70% to the training set.
- `random_state`: a seed for the random number generator used to shuffle the data before splitting it. This makes the split reproducible.
- `stratify`: This ensures that the proportion of each target class is approximately the same in both the training and test sets.

```
from sklearn.model_selection import train_test_split

X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values

X_train, X_test, y_train, y_test =\
    train_test_split(X, y,
                     test_size=0.3,
                     random_state=0,
                     stratify=y)
```

Bringing features onto the same scale

This code applies the `MinMaxScaler` method from the scikit-learn library to scale the features of the training set `X_train` and the test set `X_test`. The `MinMaxScaler` method scales each feature to a given range, usually [0, 1]. The `fit_transform` method is used to fit the scaler to the training set and also return the transformed training set. The transform method is then used to apply the same scaling parameters to the test set, so that both the training set and the test set are transformed with the same scaling parameters. This is important so that the comparison between the predicted values and the true values in the test set is fair, as both the inputs and the outputs are scaled in the same way.

```
from sklearn.preprocessing import MinMaxScaler
```

```
mms = MinMaxScaler()
X_train_norm = mms.fit_transform(X_train)
X_test_norm = mms.transform(X_test)
```

The code imports the `StandardScaler` class from scikit-learn's preprocessing module. The `StandardScaler` performs standardization on the input data, transforming it so that it has a mean of 0 and a standard deviation of 1.

Here, a `StandardScaler` object `stdsc` is created and then `fit_transform` method is called on the training data `X_train` to fit the `stdsc` object to the data and transform the training data to have a mean of `0` and a standard deviation of `1`. The same `stdsc` object is then used to transform the test data X_test with the transform method.

This is different from `MinMaxScaler` in the sense that `MinMaxScaler` scales the data to a specific range (typically 0 to 1), whereas `StandardScaler` standardizes the data so it has a mean of 0 and a standard deviation of 1.

```
from sklearn.preprocessing import StandardScaler

stdsc = StandardScaler()
X_train_std = stdsc.fit_transform(X_train)
X_test_std = stdsc.transform(X_test)
```

A visual example:

```
ex = np.array([0, 1, 2, 3, 4, 5])

print('standardized:', (ex - ex.mean()) / ex.std())

# Please note that pandas uses ddof=1 (sample standard deviation)
# by default, whereas NumPy's std method and the StandardScaler
# uses ddof=0 (population standard deviation)

# normalize
print('normalized:', (ex - ex.min()) / (ex.max() - ex.min()))
```

```
standardized: [-1.46385011 -0.87831007 -0.29277002  0.29277002  0.87831007  1.46385011]
normalized: [0.  0.2 0.4 0.6 0.8 1. ]
```

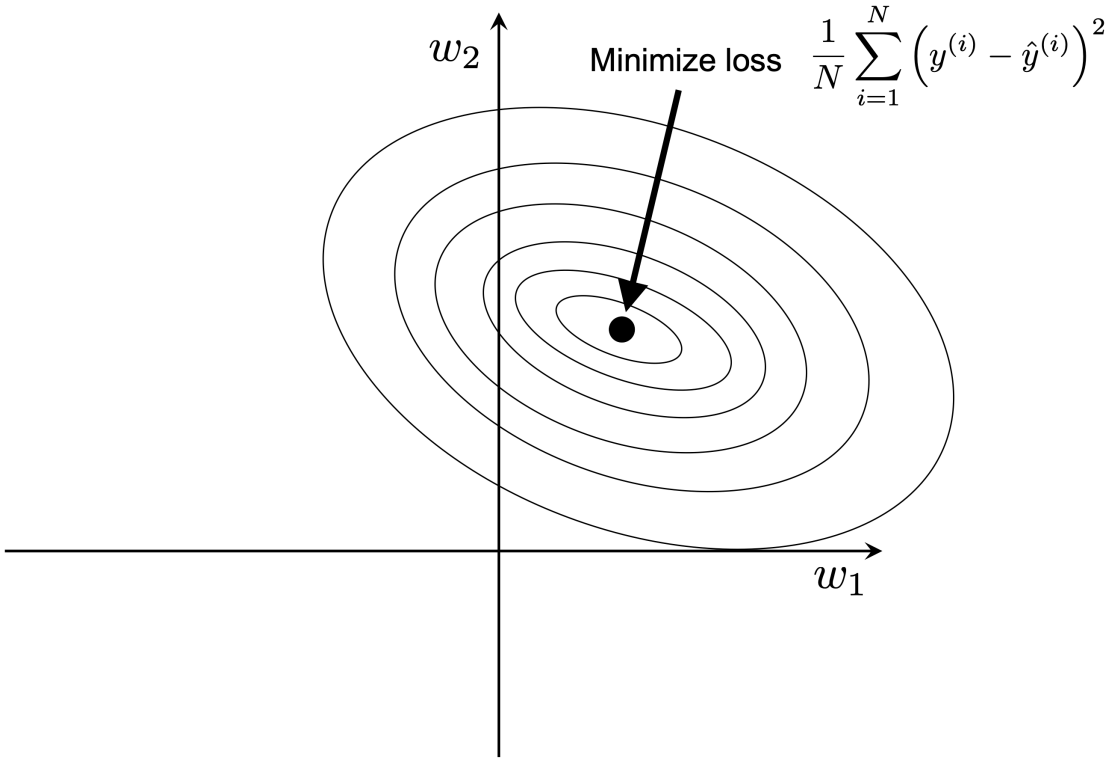
Selecting meaningful features

...

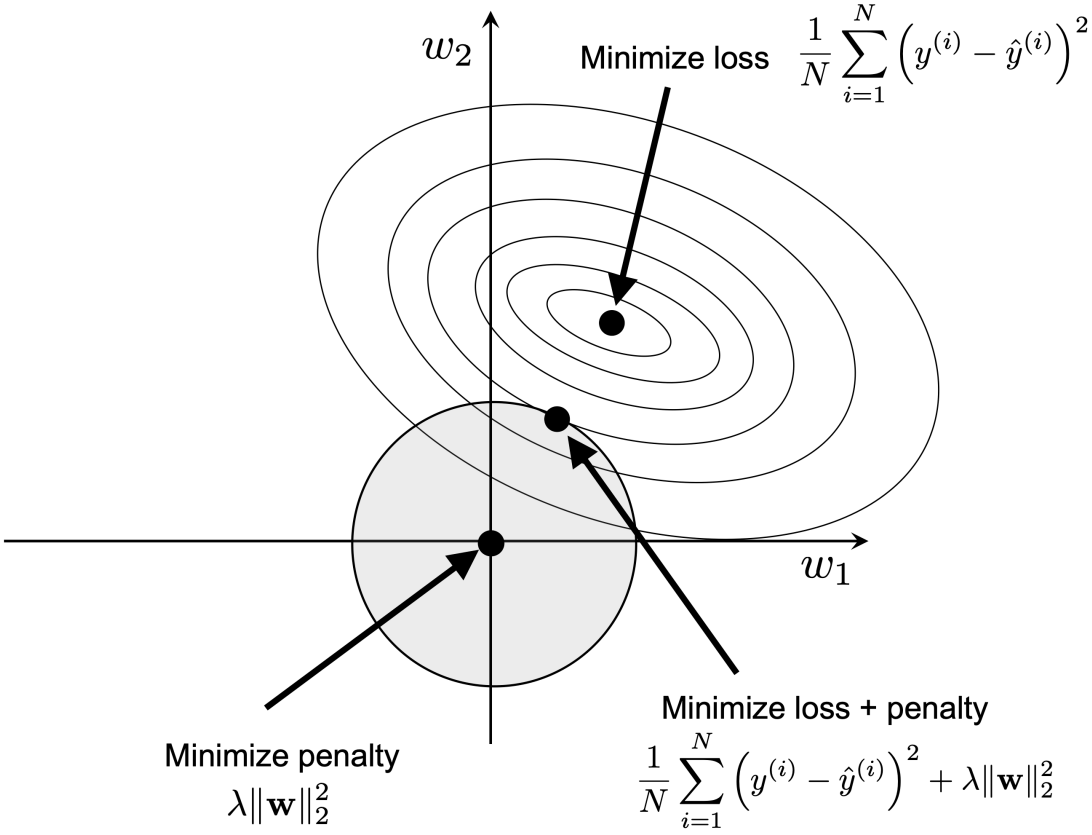
L1 and L2 regularization as penalties against model complexity

A geometric interpretation of L2 regularization

```
Image(filename='figures/04_05.png', width=500)
```

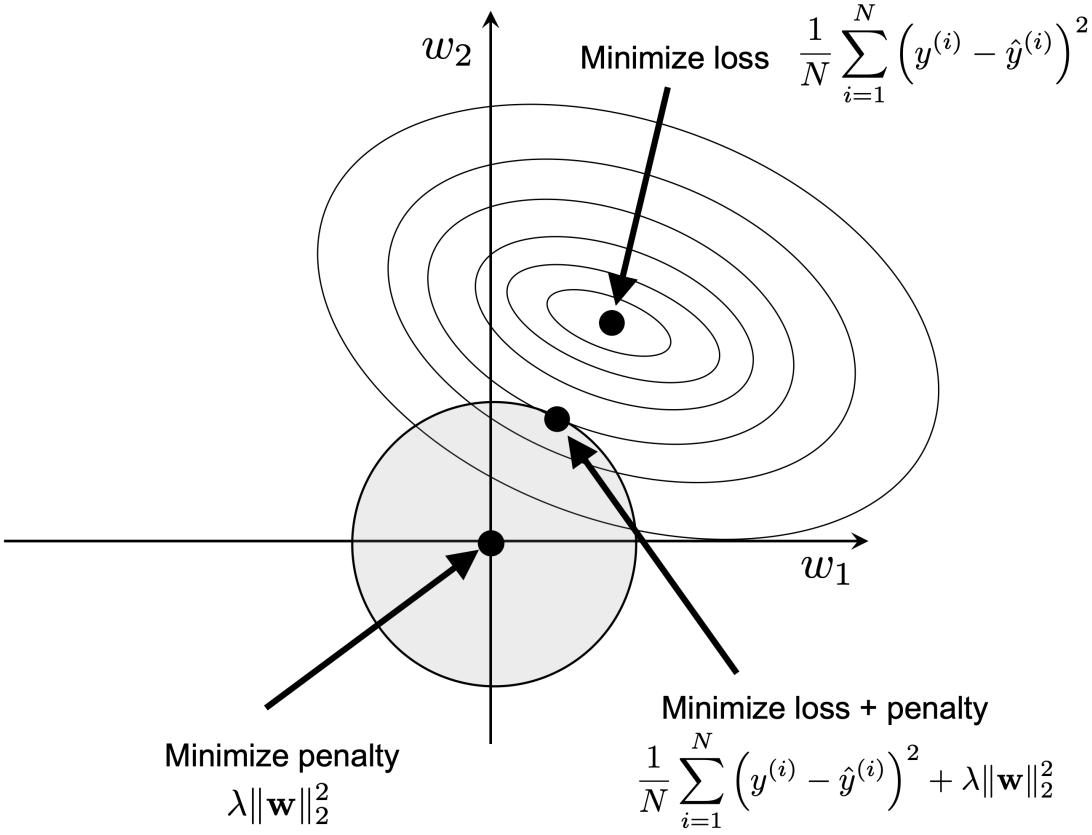



```
Image(filename='figures/04_06.png', width=500)
```



Sparse solutions with L1-regularization

```
Image(filename='figures/04_06.png', width=500)
```



For regularized models in scikit-learn that support L1 regularization, we can simply set the `penalty` parameter to `'l1'` to obtain a sparse solution:

This code imports the Logistic Regression class from the scikit-learn library.

The `LogisticRegression` class is initialized with a penalty parameter, which determines the type of regularization that is applied to the model. In this code, the penalty parameter is set to 'l1', which means that L1 regularization will be used. L1 regularization is a technique that adds a penalty term to the loss function that the model is trying to minimize. The penalty term is proportional to the absolute values of the coefficients of the features in the model. This has the effect of shrinking the magnitude of the coefficients of unimportant features towards zero, effectively reducing the dimensionality of the feature space and making the model simpler.

So, this code creates a logistic regression model with L1 regularization, which can be useful in reducing overfitting and improving the interpretability of the model.

```
from sklearn.linear_model import LogisticRegression

LogisticRegression(penalty='l1')
```

```
LogisticRegression(penalty='l1')
```

Applied to the standardized Wine data …

This code imports the `LogisticRegression` class from the scikit-learn's linear_model module. It then creates an instance of the LogisticRegression class and assigns it to the variable `lr`. The instance is configured with the L1 regularization penalty and a regularization strength of 1.0 (controlled by the C parameter). The solver `liblinear` is used for optimization and the `multi_class` parameter is set to `ovr` to handle the case of multiclass classification.

The `fit` method is then called on the `lr` object to fit the logistic regression model to the training data. The accuracy of the model is computed on both the training data and the test data using the score method and the results are printed.

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(penalty='l1', C=1.0, solver='liblinear', multi_class='ovr')
# Note that C=1.0 is the default. You can increase
# or decrease it to make the regulariztion effect
# stronger or weaker, respectively.
lr.fit(X_train_std, y_train)
print('Training accuracy:', lr.score(X_train_std, y_train))
print('Test accuracy:', lr.score(X_test_std, y_test))
```

```
Training accuracy: 1.0
Test accuracy: 1.0
```

The `lr.intercept_` attribute in this code represents the bias of the logistic regression model. Bias, also known as the intercept, is a term used in machine learning to describe the constant term that is added to the linear combination of features to make predictions. In this example, since we are using the one-versus-rest (OVR) multi-class strategy, we have three separate logistic regression models, each with its own bias term, to handle the three wine classes in the dataset. This means that for a given example, each of the three models will make a prediction, and the class label with the highest predicted probability will be chosen as the final prediction.

```
lr.intercept_
```

```
array([-1.26375788, -1.21617739, -2.3706927 ])
```

`np.set_printoptions(8)` is a line of code that sets the print options for NumPy arrays. The number "8" inside the parentheses determines the number of digits of precision to display when printing the values in a NumPy array. By setting this value to "8", we ensure that when we print NumPy arrays, they will be displayed with 8 digits of precision.

```
np.set_printoptions(8)
```

```
lr.coef_.shape
```

```
(3, 13)
```

The `lr.coef_ attribute` returns the coefficients of the logistic regression model that correspond to each input feature. The code `lr.coef_[lr.coef_!=0].shape` is checking the shape of the coefficients that are not equal to zero. This is essentially counting the number of non-zero coefficients and returning the result as a tuple with one element, the number of non-zero coefficients. This can give us an idea of how many features the model is using and whether some features are being discarded by the L1 regularization.

```
lr.coef_[lr.coef_!=0].shape
```

```
(23,)
```

```
lr.coef_
```

```
array([[ 1.24619189,  0.17996365,  0.74615176, -1.16400376,  0.        ,
         0.        ,  1.16014562,  0.        ,  0.        ,  0.        ,
         0.        ,  0.55794229,  2.50889893],
       [-1.53701968, -0.38703465, -0.9947967 ,  0.3645605 , -0.05963804,
         0.        ,  0.66791998,  0.        ,  0.        , -1.93439526,
         1.23426762,  0.        , -2.23221767],
       [ 0.13510571,  0.16929395,  0.35772024,  0.        ,  0.        ,
         0.        , -2.43428462,  0.        ,  0.        ,  1.5628059 ,
        -0.81782508, -0.49662032,  0.        ]])
```

This script demonstrates the effect of L1 regularization on feature selection in a logistic regression model. L1 regularization is a technique that can be used to reduce the complexity of a model by setting some of the weight coefficients to zero, effectively removing those features from the model (embedded feature selection tehcnique). This can be useful for improving model performance and reducing overfitting.

The script then creates a logistic regression model with L1 regularization for a range of values of the regularization strength parameter C, and fits the model to the preprocessed data. For each value of C, the script appends the weight coefficients and regularization strength to separate lists.

The script then creates a plot showing the weight coefficients for each feature as a function of the regularization strength parameter C.

The script creates a plot that visualizes the change in weight coefficients for each feature as the regularization strength is varied, with each feature plotted as a separate line in a different color. A legend is added to the plot to help identify which feature each line corresponds to. The visualization shows that as the regularization strength is increased (i.e., as C is decreased), the L1 regularization technique shrinks the weight coefficients of some of the features towards zero. This has the effect of reducing the impact of those features on the model and can be used as a technique for feature selection.

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = plt.subplot(111)

colors = ['blue', 'green', 'red', 'cyan',
          'magenta', 'yellow', 'black',
          'pink', 'lightgreen', 'lightblue',
          'gray', 'indigo', 'orange']

weights, params = [], []
for c in np.arange(-4., 6.):
    lr = LogisticRegression(penalty='l1', C=10.**c, solver='liblinear',
                            multi_class='ovr', random_state=0)
    lr.fit(X_train_std, y_train)
    weights.append(lr.coef_[1])
    params.append(10**c)

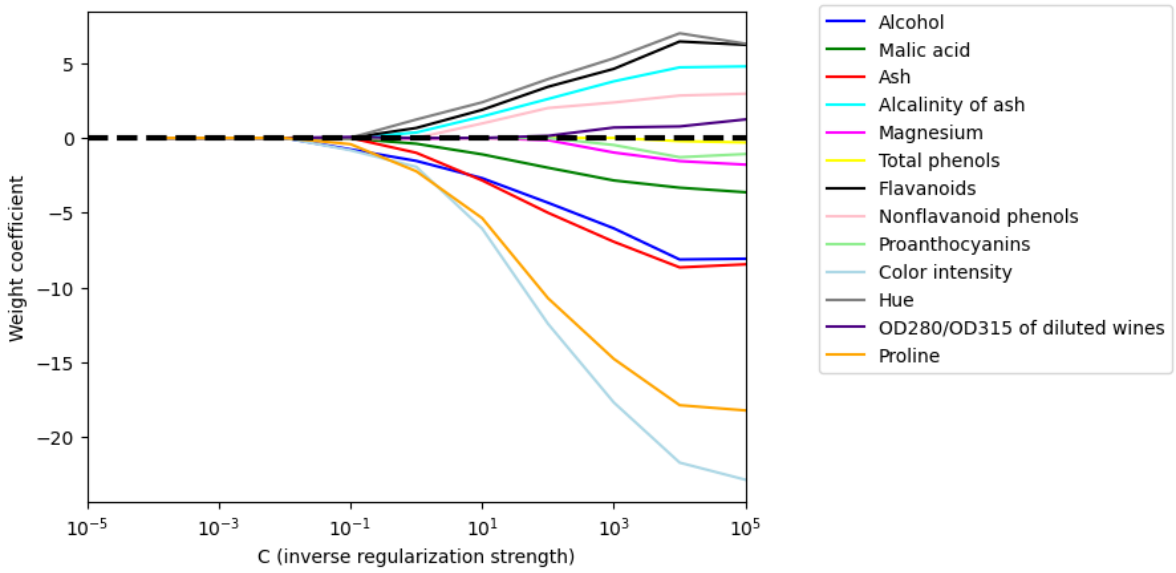
weights = np.array(weights)

for column, color in zip(range(weights.shape[1]), colors):
```

```
    plt.plot(params, weights[:, column],
             label=df_wine.columns[column + 1],
             color=color)
plt.axhline(0, color='black', linestyle='--', linewidth=3)
plt.xlim([10**(-5), 10**5])
plt.ylabel('Weight coefficient')
plt.xlabel('C (inverse regularization strength)')
plt.xscale('log')
plt.legend(loc='upper left')
ax.legend(loc='upper center',
          bbox_to_anchor=(1.38, 1.03),
          ncol=1, fancybox=True)

#plt.savefig('figures/04_08.png', dpi=300,
#            bbox_inches='tight', pad_inches=0.2)

plt.show()
```




```
weights
```

```
array([[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
```

```
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00],
[-7.66115274e-01, -4.06314474e-02,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 -8.23987196e-01,  6.97554422e-03,  6.46568870e-02,
 -4.15802417e-01],
[-1.53722377e+00, -3.87032209e-01, -9.95137081e-01,
  3.64931391e-01, -5.95585137e-02,  0.00000000e+00,
  6.67956118e-01,  0.00000000e+00,  0.00000000e+00,
 -1.93393651e+00,  1.23386892e+00,  0.00000000e+00,
 -2.23181947e+00],
[-2.69263309e+00, -1.09953732e+00, -2.84331945e+00,
  1.43730115e+00,  0.00000000e+00,  0.00000000e+00,
  1.88048254e+00,  9.69718983e-01,  0.00000000e+00,
 -6.06540983e+00,  2.38415618e+00,  0.00000000e+00,
 -5.36610108e+00],
[-4.33653058e+00, -1.99295502e+00, -5.00694069e+00,
  2.60767527e+00, -1.48022882e-01,  0.00000000e+00,
  3.42348514e+00,  2.00245197e+00,  0.00000000e+00,
 -1.24369473e+01,  3.93535103e+00,  1.37876883e-01,
 -1.07231482e+01],
[-6.05087649e+00, -2.84435070e+00, -6.94834589e+00,
  3.78883950e+00, -9.75913259e-01,  0.00000000e+00,
  4.62330715e+00,  2.37537588e+00, -4.72466879e-01,
 -1.77063485e+01,  5.33204520e+00,  6.96546239e-01,
 -1.47879004e+01],
[-8.13653733e+00, -3.32986829e+00, -8.66456824e+00,
  4.72474882e+00, -1.55129989e+00, -2.15328210e-01,
  6.45505988e+00,  2.83854129e+00, -1.29033708e+00,
 -2.17321071e+01,  7.00438310e+00,  7.70742502e-01,
 -1.78885558e+01],
[-8.08562390e+00, -3.63416028e+00, -8.45931890e+00,
```

```
       4.78819433e+00, -1.78636854e+00, -3.02528168e-01,
       6.23457801e+00,  2.95908431e+00, -1.07684960e+00,
      -2.28791425e+01,  6.30921537e+00,  1.23987802e+00,
      -1.82384248e+01]])
```

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_wine

# Load the Wine dataset
X, y = load_wine(return_X_y=True)

# Preprocess the data
scaler = StandardScaler()
X_std = scaler.fit_transform(X)

# Define a list of colors for the plot
colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black',
          'pink', 'lightgreen', 'lightblue', 'gray', 'indigo', 'orange']

# Initialize empty lists for weight coefficients and regularization strengths
weights, params = [], []

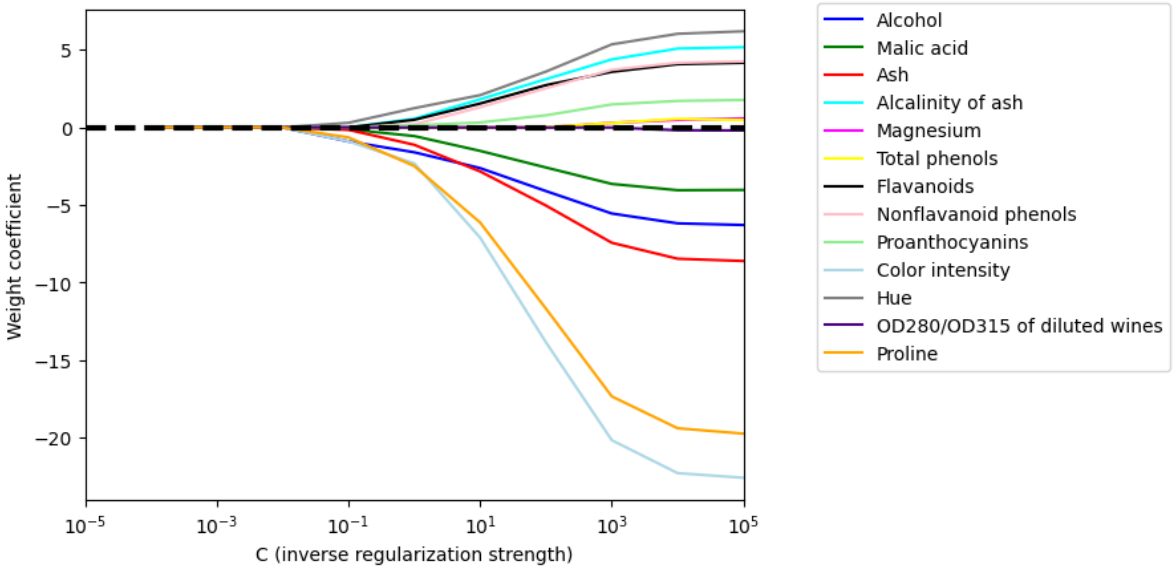
# Loop over a range of values for the regularization strength parameter C
for c in np.arange(-4., 6.):
    # Create a logistic regression model with L1 regularization
    lr = LogisticRegression(penalty='l1', C=10.**c, solver='liblinear',
                            multi_class='ovr', random_state=0)
    # Fit the model and append the weight coefficients and regularization strength
    lr.fit(X_std, y)
    weights.append(lr.coef_[1])
    params.append(10**c)

# Convert the weights list to a NumPy array
weights = np.array(weights)

# Create the plot
fig, ax = plt.subplots()
for column, color in zip(range(weights.shape[1]), colors):
```

```
    ax.plot(params, weights[:, column],
#            label=f'Feature {column+1}',
            label=df_wine.columns[column + 1],
            color=color)
ax.axhline(0, color='black', linestyle='--', linewidth=3)
ax.set_xlim([10**(-5), 10**5])
ax.set_ylabel('Weight coefficient')
ax.set_xlabel('C (inverse regularization strength)')
ax.set_xscale('log')
ax.legend(loc='upper left')
ax.legend(loc='upper center', bbox_to_anchor=(1.38, 1.03),
          ncol=1, fancybox=True)

# Show the plot
plt.show()
```



Sequential feature selection algorithms

This code defines a class called `SBS` (sequential backward selection) which is a feature selection technique for machine learning models. The class can be used to select a subset of features from a dataset that will produce a good performing model.

The `SBS` class has two main methods: `fit()` and `transform()`. The `fit()` method takes a dataset as input and performs sequential backward selection on the features, reducing the

number of features until the specified number of features is reached. The method outputs the indices of the selected features.

The `transform()` method takes the dataset as input and returns the selected subset of features.

The `SBS` class uses a provided estimator (classifier or regression model) to fit the data and a provided scoring function to evaluate the performance of the estimator. The method starts with all the features and iteratively removes the least important feature until the specified number of features is reached.

The `SBS` class first splits the dataset into training and testing sets using `train_test_split()` from scikit-learn. The method then initializes a set of indices corresponding to all the features and the set of subsets containing all the indices. The method then calculates the score of the model using the scoring function provided.

The method then loops through each feature and removes it from the feature set to create a new subset of features. The score of the new subset is calculated and stored in a list. This process is repeated until the desired number of features is reached.

The `SBS` class keeps track of the best subset of features selected and the corresponding score. Finally, the method outputs the subset of features that achieved the best score.

In summary, the `SBS` class is a useful tool for feature selection in machine learning, providing a systematic way to remove features from a dataset and select the best subset for a model.

```
from sklearn.base import clone
from itertools import combinations
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split


class SBS:
    def __init__(self, estimator, k_features, scoring=accuracy_score,
                 test_size=0.25, random_state=1):
        self.scoring = scoring
        self.estimator = clone(estimator)
        self.k_features = k_features
        self.test_size = test_size
        self.random_state = random_state

    def fit(self, X, y):

        X_train, X_test, y_train, y_test = \
```

```
            train_test_split(X, y, test_size=self.test_size,
                             random_state=self.random_state)

        dim = X_train.shape[1]
        self.indices_ = tuple(range(dim))
        self.subsets_ = [self.indices_]
        score = self._calc_score(X_train, y_train,
                                 X_test, y_test, self.indices_)
        self.scores_ = [score]

        while dim > self.k_features:
            scores = []
            subsets = []

            for p in combinations(self.indices_, r=dim - 1):
                score = self._calc_score(X_train, y_train,
                                         X_test, y_test, p)
                scores.append(score)
                subsets.append(p)

            best = np.argmax(scores)
            self.indices_ = subsets[best]
            self.subsets_.append(self.indices_)
            dim -= 1

            self.scores_.append(scores[best])
        self.k_score_ = self.scores_[-1]

        return self

    def transform(self, X):
        return X[:, self.indices_]

    def _calc_score(self, X_train, y_train, X_test, y_test, indices):
        self.estimator.fit(X_train[:, indices], y_train)
        y_pred = self.estimator.predict(X_test[:, indices])
        score = self.scoring(y_test, y_pred)
        return score
```

```
dim = 5
indices_ = tuple(range(dim))
```

```
print(indices_)

for x in combinations(indices_, r=dim - 1):
    print(x)
```

```
(0, 1, 2, 3, 4)
(0, 1, 2, 3)
(0, 1, 2, 4)
(0, 1, 3, 4)
(0, 2, 3, 4)
(1, 2, 3, 4)
```

```
type(combinations(indices_, r=dim - 1))
```

```
itertools.combinations
```

This code demonstrates how to use the `SBS` (sequential backward selection) class to select a subset of features from a dataset and evaluate the performance of a K-Nearest Neighbor (KNN) classifier on the selected subset.

First, a KNN classifier with five neighbors is initialized using `KNeighborsClassifier(n_neighbors=5)`.

Next, the `SBS` class is used to select a subset of features from the standardized training dataset `X_train_std` and the corresponding labels `y_train`. The `fit()` method of the `SBS` class is used to fit the KNN classifier to the training data and perform sequential backward selection to reduce the number of features until the desired number of features (one in this case) is reached. The resulting best subset of features is stored in the `sbs.subsets_` attribute of the `SBS` class.

Finally, a plot is created to show how the performance of the KNN classifier varies as the number of selected features is increased. The number of features in each subset is stored in the `k_feat` list, and the accuracy of the KNN classifier on each subset is stored in the sbs.scores_ list. The `plt.plot()` function is used to plot the accuracy as a function of the number of features, and various formatting functions are used to adjust the appearance of the plot.

Overall, this code provides a useful example of how to use the SBS class to select a subset of features and evaluate the performance of a machine learning model on the selected subset. The resulting plot can be used to identify the optimal number of features to use for the KNN classifier.

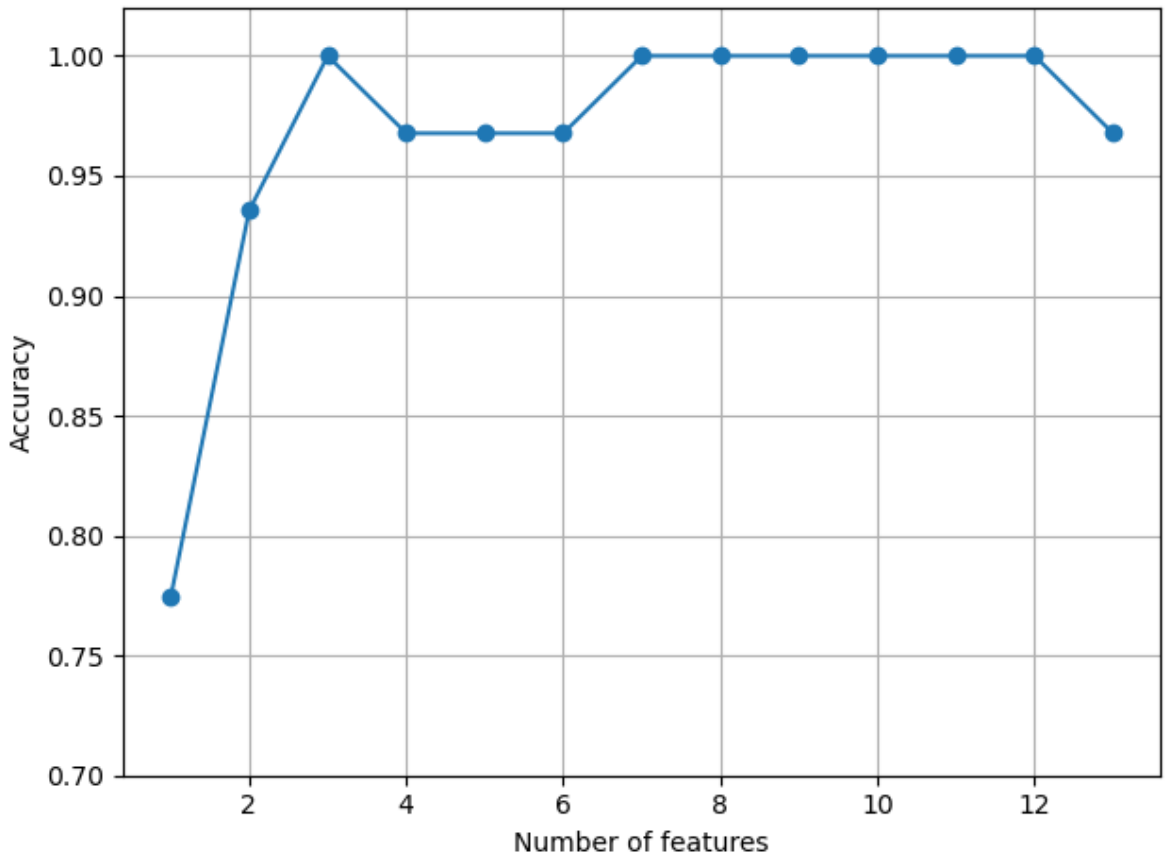
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5)

# selecting features
sbs = SBS(knn, k_features=1)
sbs.fit(X_train_std, y_train)

# plotting performance of feature subsets
k_feat = [len(k) for k in sbs.subsets_]

plt.plot(k_feat, sbs.scores_, marker='o')
plt.ylim([0.7, 1.02])
plt.ylabel('Accuracy')
plt.xlabel('Number of features')
plt.grid()
plt.tight_layout()
# plt.savefig('figures/04_09.png', dpi=300)
plt.show()
```



This code prints the names of the features that were selected by the `SBS` (sequential backward selection) class.

The code selects the 11th subset of features from the `sbs.subsets_` attribute of the SBS class, which represents the set of features selected when only 3 features remain. The indices of the selected features are stored in the `k3` list.

The `df_wine.columns[1:]` expression is used to get the names of the features, excluding the target variable column. The `k3` list is then used to select the names of the features that were selected by the `SBS` class.

Finally, the names of the selected features are printed to the console.

In summary, this code provides a way to determine which features were selected by the SBS class, which can be useful for understanding which features are most important for the KNN classifier.

```
k3 = list(sbs.subsets_[10])
print(k3)
print(df_wine.columns[1:][k3])
```

```
[0, 1, 11]
Index(['Alcohol', 'Malic acid', 'OD280/OD315 of diluted wines'], dtype='object')
```

```
knn.fit(X_train_std, y_train)
print('Training accuracy:', knn.score(X_train_std, y_train))
print('Test accuracy:', knn.score(X_test_std, y_test))
```

```
Training accuracy: 0.967741935483871
Test accuracy: 0.9629629629629629
```

```
knn.fit(X_train_std[:, k3], y_train)
print('Training accuracy:', knn.score(X_train_std[:, k3], y_train))
print('Test accuracy:', knn.score(X_test_std[:, k3], y_test))
```

```
Training accuracy: 0.9516129032258065
Test accuracy: 0.9259259259259259
```

Example

Comparing different feature seleciton technique from sciki-learn

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import RFE, RFECV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

```
# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target
```

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

```
# Select KBest features using chi2 as scoring function
kbest = SelectKBest(score_func=chi2, k=5)
X_train_kbest = kbest.fit_transform(X_train, y_train)
X_test_kbest = kbest.transform(X_test)
```

```
# Recursive Feature Elimination (RFE) with logistic regression as the estimator
rfe = RFE(estimator=LogisticRegression(), n_features_to_select=5, step=1)
X_train_rfe = rfe.fit_transform(X_train, y_train)
X_test_rfe = rfe.transform(X_test)
```

```
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```python
# Recursive Feature Elimination with Cross Validation (RFECV) with logistic regression as
rfecv = RFECV(estimator=LogisticRegression(), step=1, cv=5, scoring='accuracy')
X_train_rfecv = rfecv.fit_transform(X_train, y_train)
X_test_rfecv = rfecv.transform(X_test)
```

```
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/lin
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/lin
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/lin
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/lin
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/lin
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/bjalaian/miniconda/envs/machine-learning-book/lib/python3.9/site-packages/sklearn/line
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
# Compare the selected features by the different methods
print("Selected features by SelectKBest:", kbest.get_support(indices=True))
print("Selected features by RFE:", rfe.get_support(indices=True))
print("Selected features by RFECV:", rfecv.get_support(indices=True))
```

```
Selected features by SelectKBest: [ 2  3 13 22 23]
Selected features by RFE: [ 6 10 25 26 27]
Selected features by RFECV: [ 0  4  5  6  7  8 10 11 13 15 16 21 22 25 26 27]
```

Assessing feature importance with Random Forests

This code demonstrates how to use a random forest classifier to estimate the importance of each feature in a dataset.

First, the names of the features in the Wine dataset are assigned to the feat_labels variable.

Next, a random forest classifier is initialized with 500 trees and a random state of 1 using `RandomForestClassifier(n_estimators=500, random_state=1)`. The classifier is then fit to the training data `X_train` and the corresponding labels `y_train` using the `fit()` method.

The `feature_importances_` attribute of the random forest classifier is used to estimate the importance of each feature in the dataset. The `np.argsort()` function is used to sort the features in descending order of importance, and the sorted indices are stored in the indices variable.

Finally, a plot is created to visualize the feature importances. The `plt.bar()` function is used to create a horizontal bar plot of the feature importances, and the `plt.xticks()` function is used to label each bar with the corresponding feature name. The plot provides a visual representation of the importance of each feature, with more important features having larger bars.

Overall, this code is a useful example of how to use a random forest classifier to estimate feature importances and create a plot to visualize the results. The resulting plot can be used to identify which features are most important for the classifier and can be used to guide feature selection or engineering efforts.

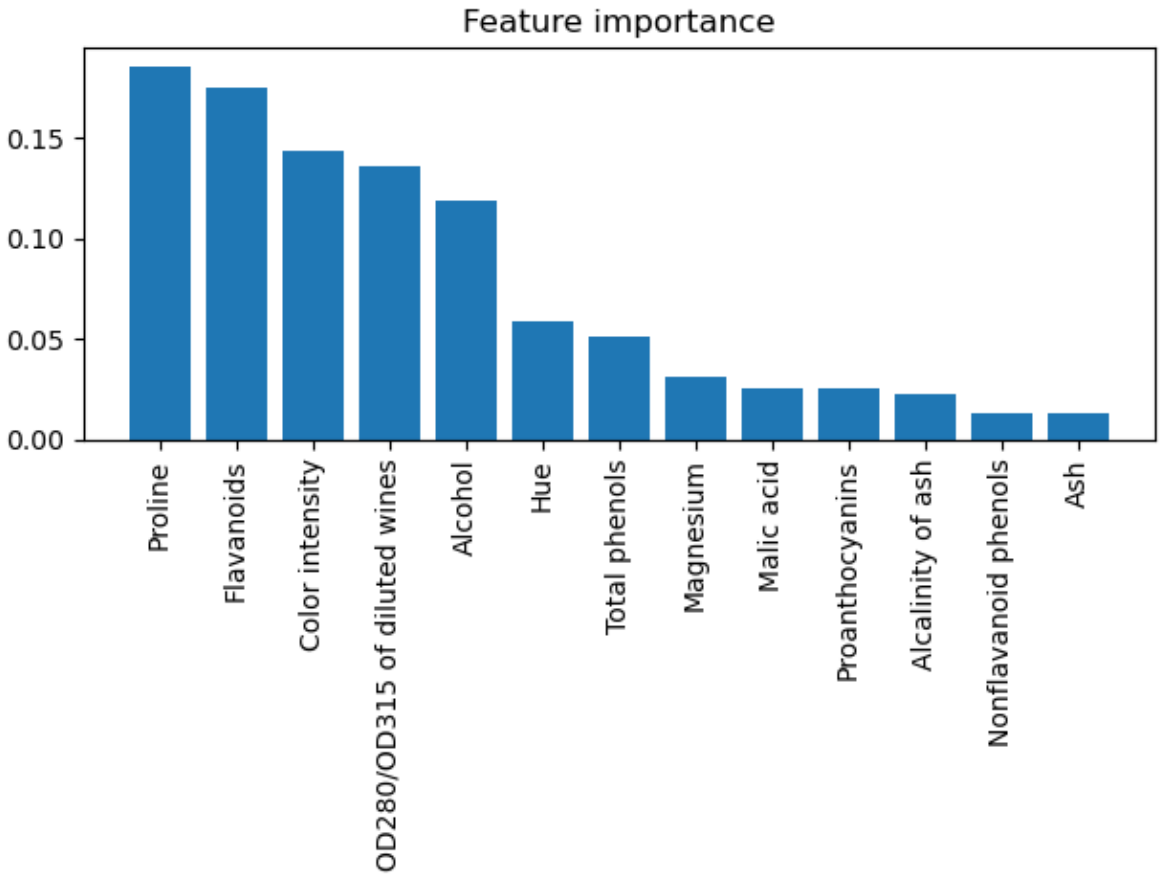
```
X_train.shape
```

```
(124, 13)
```

```
from sklearn.ensemble import RandomForestClassifier

feat_labels = df_wine.columns[1:]

forest = RandomForestClassifier(n_estimators=500,
                                random_state=1)

forest.fit(X_train, y_train)
importances = forest.feature_importances_

indices = np.argsort(importances)[::-1]

for f in range(X_train.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30,
                            feat_labels[indices[f]],
                            importances[indices[f]]))

plt.title('Feature importance')
plt.bar(range(X_train.shape[1]),
        importances[indices],
        align='center')

plt.xticks(range(X_train.shape[1]),
           feat_labels[indices], rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.tight_layout()
# plt.savefig('figures/04_10.png', dpi=300)
plt.show()
```

```
 1) Proline                        0.185453
 2) Flavanoids                     0.174751
 3) Color intensity                0.143920
 4) OD280/OD315 of diluted wines   0.136162
 5) Alcohol                        0.118529
 6) Hue                            0.058739
 7) Total phenols                  0.050872
 8) Magnesium                      0.031357
 9) Malic acid                     0.025648
10) Proanthocyanins                0.025570
11) Alcalinity of ash              0.022366
12) Nonflavanoid phenols           0.013354
```

```
13) Ash                           0.013279
```



This code demonstrates how to use the `SelectFromModel` class from scikit-learn to select a subset of features from a dataset based on a threshold of feature importance.

First, the `SelectFromModel` class is imported from scikit-learn using from `sklearn.feature_selection`.

Next, the `SelectFromModel` class is initialized with the random forest classifier forest as the estimator and a threshold of `0.1` for feature selection. The `prefit` parameter is set to `True`, which indicates that the estimator has already been fit to the data.

The `transform()` method of the `SelectFromModel` class is used to transform the training data `X_train` to the selected subset of features based on the feature importances calculated by the random forest classifier. The resulting subset of features is stored in the `X_selected` variable.

Finally, the number of features in the selected subset is printed to the console using `X_selected.shape[1]`.

In summary, this code demonstrates how to use the SelectFromModel class to select a subset of features based on a threshold of feature importance, which can be a useful technique for reducing the dimensionality of a dataset and improving the performance of machine learning models.

```
from sklearn.feature_selection import SelectFromModel

sfm = SelectFromModel(forest, threshold=0.1, prefit=True)
X_selected = sfm.transform(X_train)
print('Number of features that meet this threshold criterion:',
      X_selected.shape[1])
```

```
Number of features that meet this threshold criterion: 5
```

Now, let's print the 5 features that met the threshold criterion for feature selection that we set earlier (note that this code snippet does not appear in the actual book but was added to this notebook later for illustrative purposes):

```
for f in range(X_selected.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30,
                            feat_labels[indices[f]],
                            importances[indices[f]]))
```

```
 1) Proline                        0.185453
 2) Flavanoids                     0.174751
 3) Color intensity                0.143920
 4) OD280/OD315 of diluted wines   0.136162
 5) Alcohol                        0.118529
```

Summary

…

Readers may ignore the next cell.

```
! python ../.convert_notebook_to_script.py --input ch04.ipynb --output ch04.py
```

```
[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not recognized by `NbConve
[NbConvertApp] Converting notebook ch04.ipynb to script
[NbConvertApp] Writing 16590 bytes to ch04.py
```