All Modules Used In The Code

- **pandas**: This library provides data structures and functions needed for manipulating structured data. It provides two primary data structures, Series(1-dimensional) and DataFrame(2-dimensional), for handling data. These data structures can hold any type of data.

- **numpy**: This library provides support for arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays. It's used for scientific computing in Python.

- **nltk (Natural Language Toolkit)**: This is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

- **re (Regular expression operations)**: This module provides regular expression matching operations similar to those found in Perl. Regular expressions use the backslash character ('') to indicate special forms or to allow special characters to be used without invoking their special meaning.

- **string**: This module contains various string constant which contains the ASCII characters of all cases. String constants are used in operations like string formatting.

- **wordcloud**: A word cloud is a novelty visual representation of text data. It is useful for quickly perceiving the most prominent terms and for locating a term alphabetically to determine its relative prominence.

- **PorterStemmer**: This is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalisation process that is usually done when setting up Information Retrieval systems.

- **TfidfVectorizer**: TF-IDF stands for "Term Frequency, Inverse Document Frequency." It's a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents. If a word appears frequently in a document, it's important. If a word appears in many documents, it's not a unique identifier.

- **sklearn (Scikit-Learn)**: This is one of the most widely used machine learning libraries. It contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

- **SVC (Support Vector Classifier)**: The objective of a Linear SVC (Support Vector Classifier) is to fit to the data provided, returning a "best fit" hyperplane that divides, or categorizes, the data. After getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.

- **LabelEncoder**: This is a utility class to help normalize labels such that they contain only values between 0 and n_classes-1. It can also be used to transform non-numerical labels to numerical labels.

- **StandardScaler**: StandardScaler standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation.

- **MinMaxScaler**: This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

- **ExtraTreesClassifier**: This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

- **make_pipeline**: This is a utility function for making pipelines. The function collects all the steps in the pipeline, ensuring that all steps are checked for validity before proceeding. Finally, it builds and returns a composite estimator.

- **GridSearchCV**: This is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.

- **LogisticRegression**: Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

- **DecisionTreeClassifier**: Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

- **RandomForestClassifier**: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

- **BernoulliNB**: It implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable.

- **KNeighborsClassifier**: This classifier implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user.

- **OneVsRestClassifier**: This strategy, also known as one-vs-all, is implemented in OneVsRestClassifier. The strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.

- **Pipeline**: Pipeline can be used to chain multiple estimators into one. This is useful as there is often a fixed sequence of steps in processing the data, for example feature selection, normalization and classification.

- **train_test_split**: This function is a quick utility that wraps input validation and next(ShuffleSplit().split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

- **label_binarize**: This function binarizes labels in a one-vs-all fashion. Several regression and binary classification algorithms are available in the scikit. A simple way to extend these algorithms to the multi-class classification case is to use the so-called one-vs-all scheme.

- **svm (Support Vector Machines)**: Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

- **datasets**: This package embeds some small toy datasets as introduced in the Getting Started section.

- **preprocessing**: The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

- **metrics**: The sklearn.metrics module includes score functions, performance metrics and pairwise metrics and distance computations.

- **classification_report**: Build a text report showing the main classification metrics.

- **cross_val_score**: Evaluate a score by cross-validation.

- **roc_auc_score**: Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

- **roc_curve**: Compute Receiver operating characteristic (ROC).

- **auc**: Compute Area Under the Curve (AUC) using the trapezoidal rule.

- **matplotlib.pyplot**: This is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

- **rcParams**: This is a dictionary object that contains default settings for matplotlib. You can modify the settings in this dictionary to customize matplotlib.

- **seaborn**: This is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- **TextBlob**: This is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

- **tools from plotly**: This module provides tools for creating subplot grids, managing subplot layout, and creating figures with subplots.

- **graph_objs from plotly**: This module provides classes for creating figures and layouts.

- **iplot from plotly.offline**: This function creates a plotly graph inside an IPython notebook without connecting to an external server.

- **%matplotlib inline**: This is a magic function in IPython. This function allows the output of plotting commands to be displayed inline within frontends like the Jupyter notebook, directly below the code cell that produced it.

- **warnings**: This module is used to warn the user of some condition in the program, where that condition (normally) doesn't warrant raising an exception and terminating the program.

- **interp from scipy**: This function is used for one-dimensional linear interpolation.

- **cycle from itertools**: This function returns elements from the iterable until it is exhausted. Then it repeats the sequence indefinitely.

- **cufflinks**: This library binds the power of plotly with the flexibility of pandas for easy plotting.

- **defaultdict from collections**: This is a dictionary subclass that calls a factory function to supply missing values.

- **Counter from collections**: This is a dict subclass for counting hashable objects.

- **SMOTE from imblearn.over_sampling**: This function is used to handle the imbalanced dataset. It works by creating synthetic samples from the minor class instead of creating copies. This is done until the majority and minority class becomes balanced out.