7 Types of Classification Algorithms

Rohit Garg
f2005636@gmail.com

Abstract: The purpose of this research is to put together the 7 most commonly used classification algorithms along with the python code: Logistic Regression, Naïve Bayes, Stochastic Gradient Descent, K-Nearest Neighbours, Decision Tree, Random Forest, and Support Vector Machine

1 Introduction

1.1 Structured Data Classification

Classification can be performed on structured or unstructured data. Classification is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under.

- **Classifier:** An algorithm that maps the input data to a specific category.
- **Feature:** A feature is an individual measurable property of a phenomenon being observed.
- **Feature selection:** It is the process of identifying/deriving the most meaningful data (features) from the given input.
- **Classification model:** A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.
- **Binary Classification:** Classification task with two possible outcomes. Eg: Gender classification (Male / Female)
- **Multi class classification:** Classification with more than two classes. In multi class classification each sample is assigned to one and only one target label. Eg: An animal can be cat or dog but not both at the same time
- **Multi label classification:** Classification task where each sample is mapped to a set of target labels (more than one class). Eg: A news article can be about sports, a person, and location at the same time.
- **Supervised classification:** It is a technique where the learning is based on a training set of correctly labelled observations. Eg: Email classification where input data is a set of emails labeled as spam/not spam.
- **Unsupervised classification:** Grouping the observations into various categories based on some similarity measures. Eg: Grouping of news articles based on the content.

The following are the steps involved in building a classification model:

- **Initialize** the classifier to be used.
- **Train the classifier:** All classifiers in scikit-learn uses a fit(X, y) method to fit the model(training) for the given train data X and train label y.
- **Predict the target:** Given an unlabeled observation X, the predict(X) returns the predicted label y.
- **Evaluate** the classifier model

1.2 Dataset Source and Contents

This data was extracted from the census bureau database found at

- **http://www.census.gov/ftp/pub/DES/www/welcome.html**
- **Total**: 48,842 instances
- **Train**: 32,561 instances
- **Test**: 16,281 instances

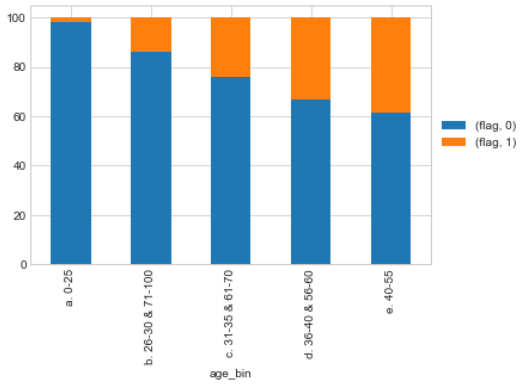
The dataset contains salaries. The following is a description of our dataset:

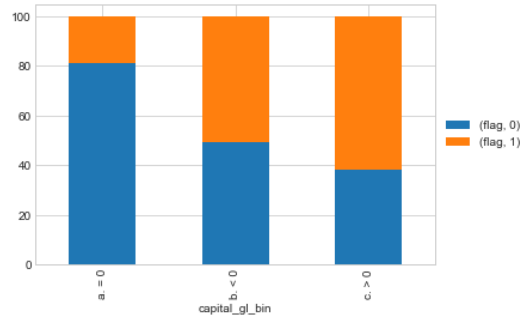
- **No. of Classes:** 2 ('>50K' and '<=50K')
- **No. of attributes (Columns):** 7
- **No. of instances (Rows):** 48,842

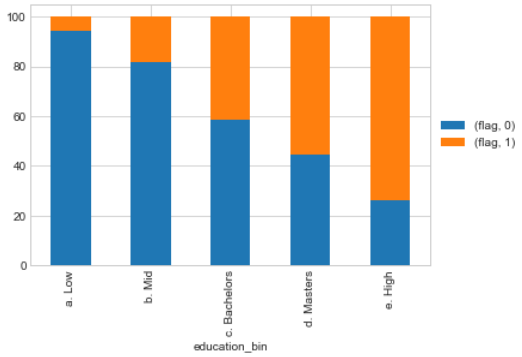
1.3 Exploratory Data Analysis

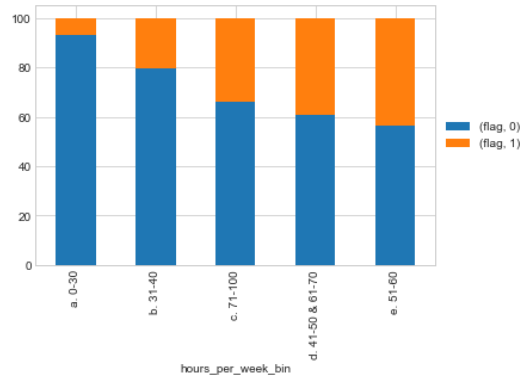
There are 7 explanatory variables:

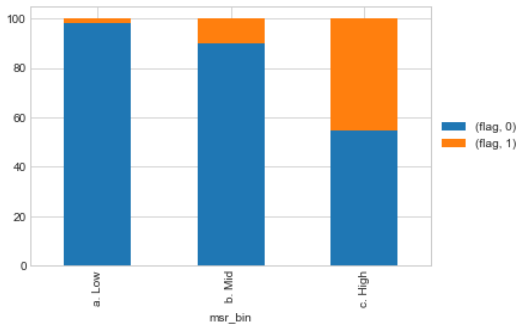
1. **Age (5 bins)**
2. **Capital Gain / Loss (3 bins)**
3. **Education (5 bins)**
4. **Hours per Week (5 bins)**
5. **Marriage Status and Relationship (3 bins)**
6. **Occupation (5 bins)**
7. **Race and Sex (3 bins)**

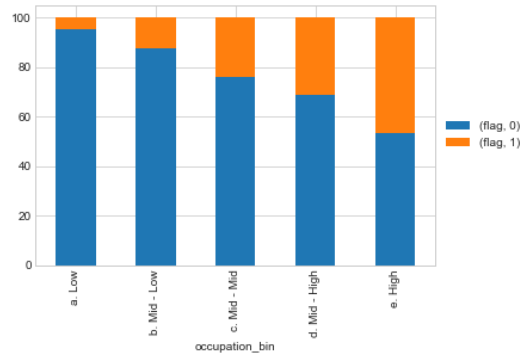


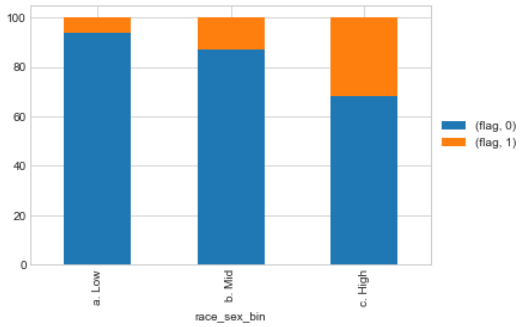












2 Classification Algorithms (Python)

2.1 Logistic Regression

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function. The implementation of logistic regression in scikit-learn can be accessed from class Logistic Regression. This implementation can fit binary, One-vs-Rest, or multinomial logistic regression with optional L2 or L1 regularization.

```
from sklearn.linear_model import LogisticRegression
lr =  LogisticRegression()
lr.fit(x_train, y_train)
y_pred=lr.predict(x_test)
```

2.2 Naïve Bayes

Gaussian Naïve Bayes implements the Gaussian Naive Bayes algorithm for classification. In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

```
from sklearn.naive_bayes import GaussianNB
nb =  GaussianNB()
nb.fit(x_train, y_train)
y_pred=nb.predict(x_test)
```

2.3 Stochastic Gradient Descent

Stochastic gradient descent is a simple yet very efficient approach to fit linear models. It is particularly useful when the number of samples (and the number of features) is very large. It supports different loss functions and penalties for classification.

```
from sklearn.linear_model import SGDClassifier
sgd =  SGDClassifier(loss='modified_huber', shuffle=True,random_state=101)
sgd.fit(x_train, y_train)
y_pred=sgd.predict(x_test)
```

2.4 K-Nearest Neighbours

Neighbours-based classification is a type of instance-based learning or non-generalizing learning. It does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbours of each point: a query point is assigned the data class which has the most representatives within the nearest neighbours of the point. The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=15)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
```

2.5 Decision Tree

Decision Tree Classifier is a class capable of performing multi-class classification on a dataset. As with other classifiers, Decision Tree Classifier takes as input two arrays: an array X, sparse or dense, of size [n_samples, n_features] holding the training samples, and an array Y of integer values, size [n_samples], holding the class labels for the training samples. Decision Tree Classifier is one of the commonly used classification technique for performing binary as well as multi class classification. The decision tree model predicts the class/target by learning simple decision rules from the features of the data.

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=10, random_state=101, max_features = None, min_samples_leaf = 15)
dtree.fit(x_train, y_train)
y_pred=dtree.predict(x_test)
```

2.6 Random Forest

In random forests each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

```
from sklearn.ensemble import RandomForestClassifier
rfm = RandomForestClassifier(n_estimators=70, oob_score=True, n_jobs=-1, random_state=101, max_features = None, min_samples_leaf = 30)
rfm.fit(x_train, y_train)
y_pred=rfm.predict(x_test)
```

2.7 Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are: effective in high dimensional spaces, effective in cases where number of dimensions is greater than the number of samples, it is also memory efficient, and it is versatile.

```
from sklearn.svm import SVC
svm = SVC(kernel="linear", C=0.025,random_state=101)
svm.fit(x_train, y_train)
y_pred=svm.predict(x_test)
```

3 Conclusion

3.1 Comparison Matrix

- **Accuracy: (True Positive + True Negative) / Total Population**
  - True Positive: The number of correct predictions that the occurrence is positive
  - True Negative: The number of correct predictions that the occurrence is negative
- **F1-Score: (2 x Precision x Recall) / (Precision + Recall)**
  - Precision: When a positive value is predicted, how often is the prediction correct?
  - Recall: When the actual value is positive, how often is the prediction correct?

Classification Algorithms	Accuracy	F1-Score
Logistic Regression	84.60%	0.6337
Naïve Bayes	80.11%	0.6005
Stochastic Gradient Descent	82.20%	0.5780
K-Nearest Neighbours	83.56%	0.5924
Decision Tree	84.23%	0.6308
Random Forest	84.33%	0.6275
Support Vector Machine	84.09%	0.6145

Code location: https://github.com/f2005636/Classification

3.2 Algorithm Selection