Higher National Diploma in Data Science – 21.1F

Module – Machine Learning

Year: 02

CARS - PURCHASE DECISION MODEL
(Assignment 01)

By

PRABHASHI MAHAWATTA

COHNDDS(F/T)21.1F-017

BSc(hons.) Data Science

Business Analytics Center

National Innovation Center

August

01.<u>Introduction</u>

Using Random Forest Classification, construct a binary classification model to forecast the purchase of an automobile.

02.<u>Binary Classification</u>

Binary classification refers to those classification task that have two class labels.

Example: Purchase decision (purchase or not)

Typically, binary classification tasks involve one class that is the normal state and another class that is the abnormal state. For example “*not purchase*” is the normal state and “*purchase*” is the abnormal state. Another example is “*cancer not detected*” is the normal state of a task that involves a medical test and “*cancer detected*” is the abnormal state. The class for the normal state is assigned the class label 0 and the class with the abnormal state is assigned the class label 1.

It is common to model a binary classification task with a model that predicts a Bernoulli probability distribution.
The Bernoulli distribution is a discrete probability distribution that covers a case where an event will have a binary outcome as either a 0 or 1. For classification, this means that the model predicts a probability of an example belonging to class 1, or the abnormal state.

03.<u>DataSet</u>

Name: Cars – Purchase Decision

About Dataset:

A purchase decision data set, including whether or not a client bought a car. This dataset contains details of 1000 customers who intend to buy a car, considering their annual salaries.

Columns:

- User ID
- Gender
- Age
- Annual Salary
- Purchase Decision (No = 0, Yes = 1)

04.Understanding Data

Columns	Data Type
User ID	Integer
Gender	Object
Age	Integer
Annual Salary	Integer
Purchase Decision	Integer

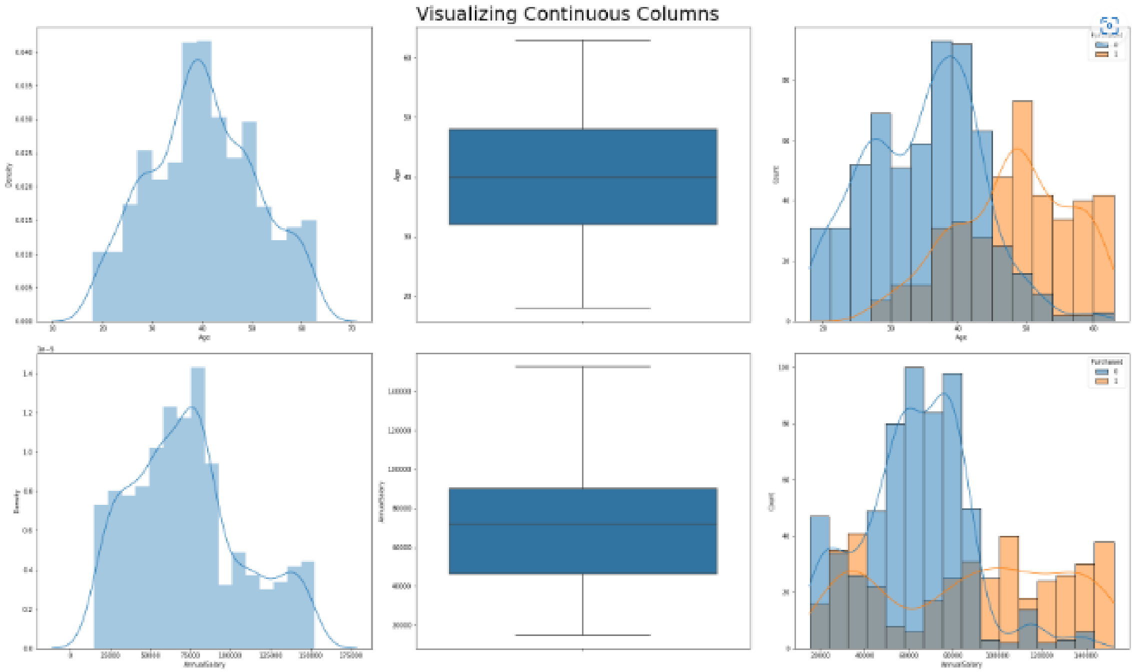
```
data.describe()
```

	User ID	Age	AnnualSalary	Purchased
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	40.106000	72689.000000	0.402000
std	288.819436	10.707073	34488.341867	0.490547
min	1.000000	18.000000	15000.000000	0.000000
25%	250.750000	32.000000	46375.000000	0.000000
50%	500.500000	40.000000	72000.000000	0.000000
75%	750.250000	48.000000	90000.000000	1.000000
max	1000.000000	63.000000	152500.000000	1.000000

Observations:

User ID – Should drop this column as it won't help the model.
Age – Minimum-18 years, Maximum- 63 years, Mean is nearly equal to Median (Normal distribution.)
AnnualSalary – Minimum – 15000, Maximum -152500

Visualization



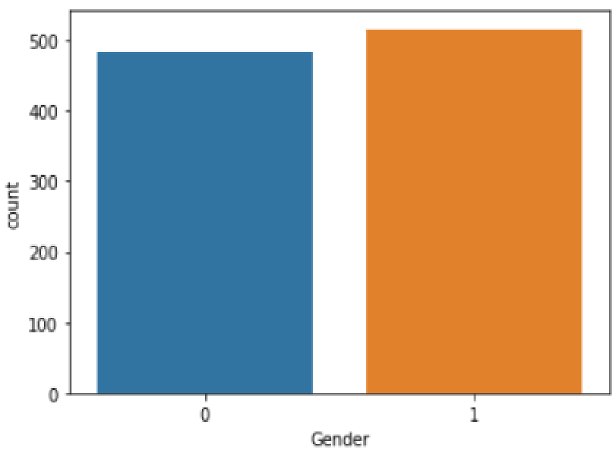
Observations:

Age

- No outline present in the dataset.
- Normally Distributed. People of Age>45 usually tend to Purchase the car.

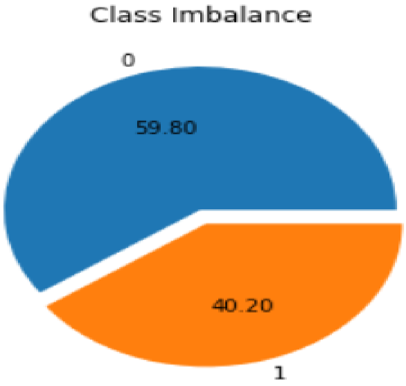
Annual Salary

- No Outliners present in the dataset.
- Bit Right Skewed. People tend to buy Car regardless of their annual Salary. Although people with salary range - 40k to 85k tend to not to purchase car.



Observations:

The dataset contains more samples with Female candidates.

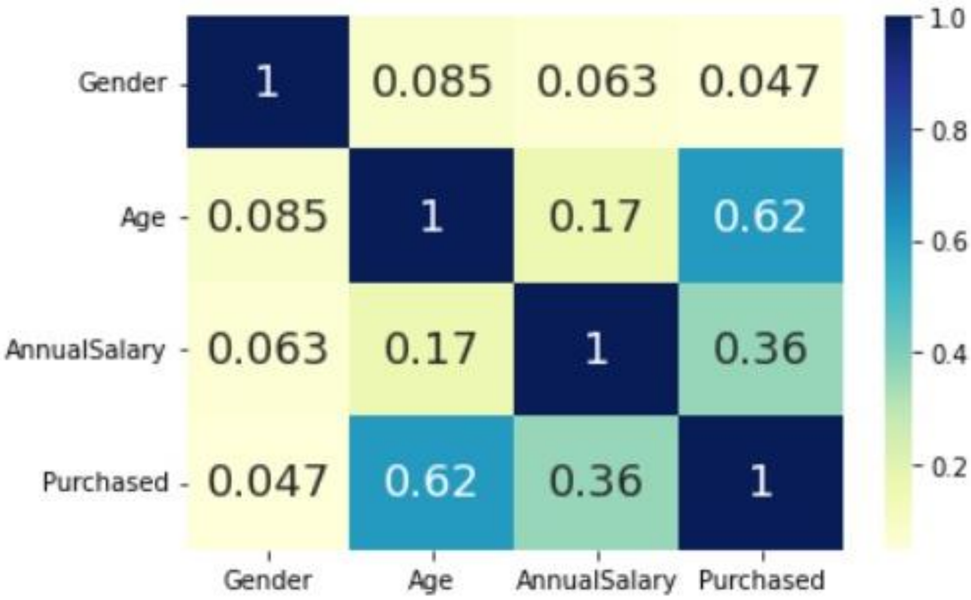


Observations:
Car was not purchases were higher in number.

Correlation And HeatMap

```
hm = data.corr()
hm
```

	Gender	Age	AnnualSalary	Purchased
Gender	1.000000	0.084760	0.063301	0.047211
Age	0.084760	1.000000	0.166042	0.616036
AnnualSalary	0.063301	0.166042	1.000000	0.364974
Purchased	0.047211	0.616036	0.364974	1.000000



05.Data Pre-Processing

Checking null values.

```
data.isna().any()
```

```
User ID         False
Gender          False
Age             False
AnnualSalary    False
Purchased       False
dtype: bool
```

Data Cleaning.

```
data = data.drop(columns = "User ID")
```

Transformation.

```
data['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

```
data['Gender'] = data['Gender'].str.replace('Male', '0')
data['Gender'] = data['Gender'].str.replace('Female', '1')
data['Gender'] = data['Gender'].astype(int)
```

06.Classification Model

I used Random Forest Classification and Random Over Sample as classification models for predictions.

I. Random Forest Classification

Random forest is an ensemble machine learning algorithm.

It is perhaps the most popular and widely used machine learning algorithm given its good or excellent performance across a wide range of classification and regression predictive modeling problems. It is also easy to use given that it has few key hyperparameters and sensible heuristics for configuring these hyperparameters.

Randomness is used in the construction of the model. This means that each time the algorithm is run on the same data, it will produce a slightly different model.

When using machine learning algorithms that have a stochastic learning algorithm, it is good practice to evaluate them by averaging their performance across multiple runs or repeats of cross-validation. When fitting a final model, it may be desirable to either increase the number of trees until the variance of the model is reduced across repeated evaluations, or to fit multiple final models and average their predictions.

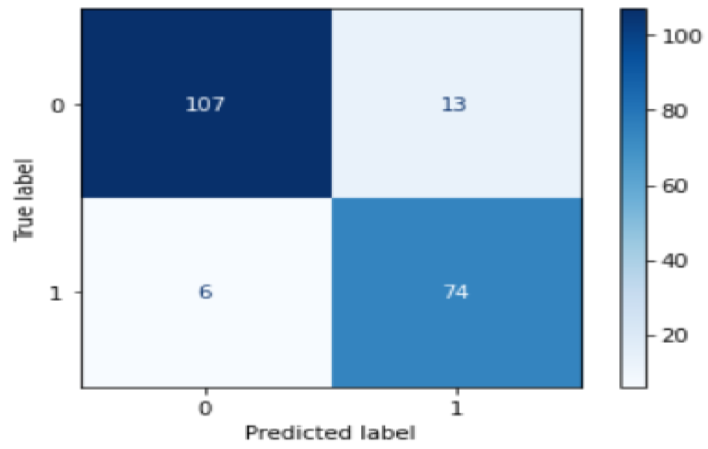
Model

Mean Absolute Error: 0.095
Accuracy: 0.91
Precision Score: 0.8505

Classification Report

	precision	recall	f1-score	support
0	0.95	0.89	0.92	120
1	0.85	0.93	0.89	80
accuracy			0.91	200
macro avg	0.90	0.91	0.90	200
weighted avg	0.91	0.91	0.91	200

Confusion Matrix



II. <u>Random Over Sampling</u>

Random oversampling involves randomly duplicating examples from the minority class and adding them to the training dataset.

Examples from the training dataset are selected randomly with replacement. This means that examples from the minority class can be chosen and added to the new "*more balanced*" training dataset multiple times; they are selected from the original training dataset, added to the new training dataset, and then returned or "*replaced*" in the original dataset, allowing them to be selected again.

It might be useful to tune the target class distribution. In some cases, seeking a balanced distribution for a severely imbalanced dataset can cause affected algorithms to over fit the minority class, leading to increased generalization error. The effect can be better performance on the training dataset, but worse performance on the holdout or test dataset.

Model

Data Reshape

```
# Reshape the data
X, y = make_classification(n_classes = 2, class_sep = 2, weights = [0.598, 0.402],
                           n_informative = 3, n_redundant = 1, flip_y = 0, n_features = 20,
                          n_clusters_per_class = 1, n_samples = 1000, random_state = 10)

print('Orignal dataset shape %s' % Counter(y))
```

```
Orignal dataset shape Counter({0: 598, 1: 402})
```

```
ros = RandomOverSampler(random_state = 42)
X_res, y_res = ros.fit_resample(X, y)

print('Reshaped dataset shape %s' % Counter(y_res))
```

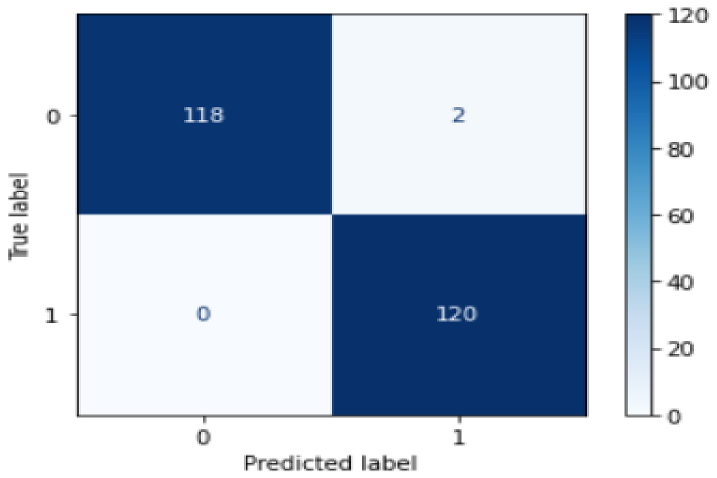
```
Reshaped dataset shape Counter({0: 598, 1: 598})
```

Accuracy: 0.99167

Classification Report

	precision	recall	f1-score	support
0	1.00	0.98	0.99	120
1	0.98	1.00	0.99	120
accuracy			0.99	240
macro avg	0.99	0.99	0.99	240
weighted avg	0.99	0.99	0.99	240

Confusion Matrix



07.Conclusion

The 99% accuracy is obtained by Random Forest Classifier trained using data from RandomOverSampler or Oversampling, but over sampling can also lead to overfitting sometimes.