# Decision Tree and Random Forest

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

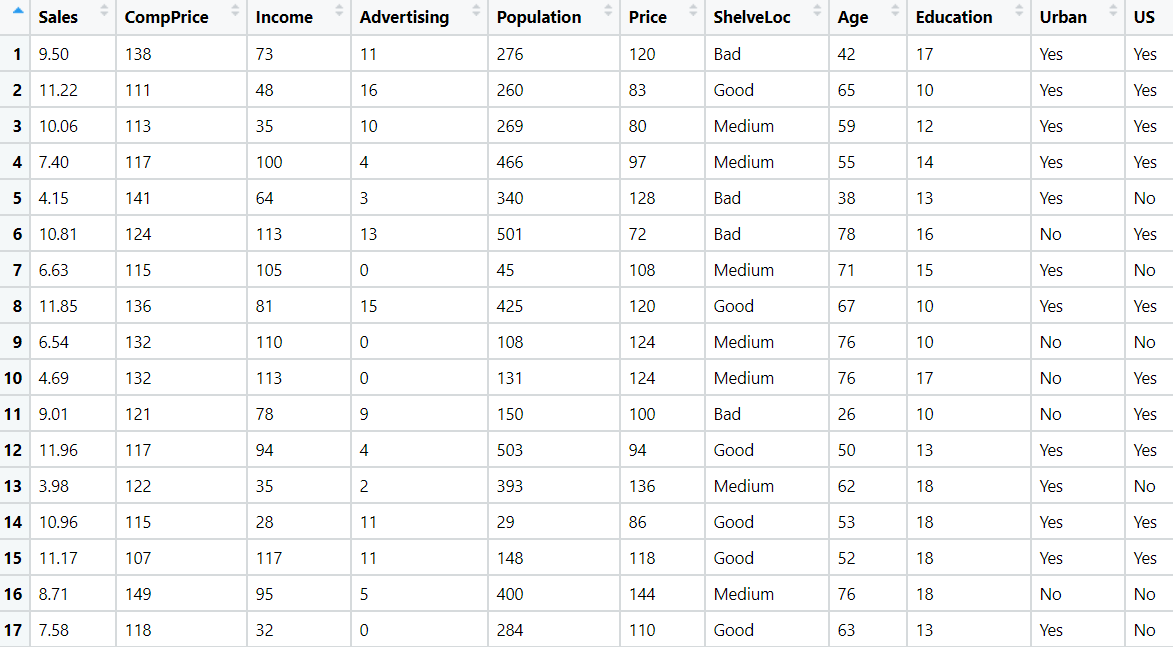
Please ensure you update all the details:

**Name: \_Prajay B. Urkude Batch ID:** \_**16092021**

**Topic: Decision Tree and Random Fores****t**

**Problem Statements:**

1. A cloth manufacturing company is interested to know about the different attributes contributing to high sales. Build a decision tree & random forest model with Sales as target variable (first convert it into categorical variable).



**ANS :- Business Objectives :-**

To know about the segment or attributes causes high sale.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| Sales | Unit sales at each location | Quantitative, Ratio | Relevant |
| CompPrice | Price charged by competitor at each location | Quantitative, Ratio | Relevant |
| Income | Community income level | Quantitative, Ratio | Relevant |
| Advertising | Local advertising budget for company at each location | Quantitative, Ratio | Relevant |
| Population | Population size in region | Quantitative, Ratio | Relevant |
| Price | Price company charges for cloth at each site | Quantitative, Ratio | Relevant |
| Shelve Location | the quality of the shelving location | Qualitative, ordinal | Relevant |
| Age | Average age of the local population | Quantitative, Ratio | Relevant |
| Education | Education level at each location | Quantitative, Ratio | Relevant |
| Urban | whether the store is in an urban or rural location | Qualitative Nominal | Relevant |
| US | whether the store is in the US or not | Qualitative, Nominal | Relevant |

* Import the libraries and different packages like pandas, numpy, matplotlib, sklearn.
* From sklearn library and tree package we import DecisionTreeClassifier function.It is used to create a classification model by building decision tree.
* From sklearn library and ensemble package import RandomForestClassifier function.
* 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.
* From sklearn library and model\_selction package import GridSearchCV function which is widely used for hyperparameter tuning to control the overfitting of the model.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e. top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e. datatypes of the datasets
* Exploratory data analysis. mean, mediam, mode, count, min max etc.
* Droppimg the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Converting the continuous data into discrete form.
* Splitting the data into train and test datasets
* Initializing the DecisionTreeClassifier model and fit the training data into it and evaluate the model on test datasets and checking the accuracy and again evaluate the same model on train datasets and compare the accuracy.

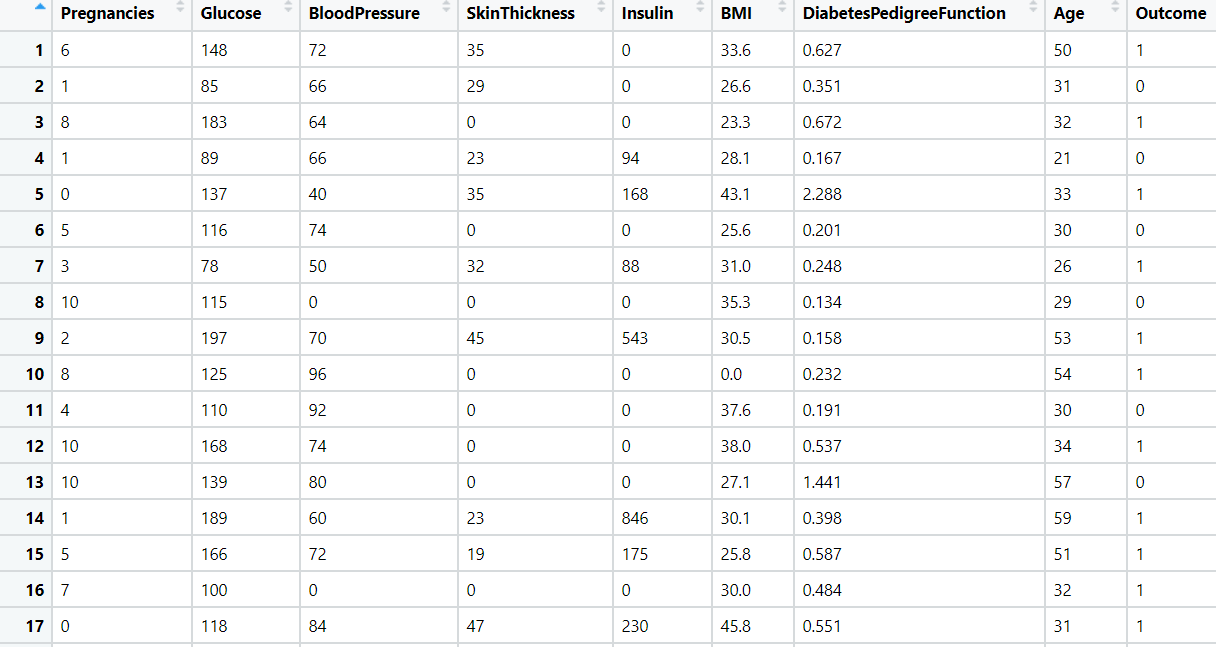
In our calculation we found that test accuracy is 62% and the train accuracy is 100% so the model is overfit model

* To overcome the overfitting we use the RandomForestClassifier technique.
* Here we initialize the function by adding the different parameters like n\_features i.e no. of trees we want to make , n\_jobs, random\_state etc.and fit the training data into it and evaluate this model into test data and again on the training data.

In our calculation we found that the test accuracy is 73% and train accuracy is 99% which is still overfit model but slightly better than previous model.

* Again we use GridSearchCv model to overcome the overfit model. Here we add the parameters Randoforestclassifier parameters and different hyperparameters like cv i.e cross validation, max\_features, min\_samples\_split to get the accuracy in the model and then initialize the GridSearchCV model and fit it into train datasets.
* We find best parameters and by taking this parameters we evaluate the model on the test datasets and again we evaluate the model on train datasets and compare the accuracy.

In our calculation we found that the test accuracy is 75% and train accuracy is 100% which is still overfit model but test accuracy is slightly greater than previous model.



1. Divide the diabetes data into train and test datasets and build a Random Forest and Decision Tree model with Outcome as the output variable.

**Business Objectives :-**

To create the classification model to predict the person having diabetes or not.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| Number of times pregnant | Number of times pregnant | Quantitative, Ratio | Relevant |
| Plasma glucose concentration | Plasma glucose concentration a 2hours in an oral glucose tolerance test | Quantitative, Ratio | Relevant |
| Diastolic blood pressure | Diastolic blood pressure (mm Hg) | Quantitative, Ratio | Relevant |
| Triceps skin fold thickness | Triceps skin fold thickness (mm) | Quantitative, Ratio | Relevant |
| 2-Hour serum insulin | 2-Hour serum insulin (mu U/ml) | Quantitative, Ratio | Relevant |
| Body mass index | Body mass index (weight in kg/(height in m)^2) | Quantitative, Ratio | Relevant |
| Diabetes pedigree function | Diabetes pedigree function | Quantitative, Ratio | Relevant |
| Age (years) | Age (years) | Quantitative, Ratio | Relevant |
| Class | Class variable (0 or 1) | Quantitative, Nominal | Relevant |



* Import the libraries and different packages like pandas, numpy, matplotlib, sklearn.
* From sklearn library and tree package we import DecisionTreeClassifier function.It is used to create a classification model by building decision tree.
* From sklearn library and ensemble package import RandomForestClassifier function.
* 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.
* From sklearn library and model\_selction package import GridSearchCV function which is widely used for hyperparameter tuning to control the overfitting of the model.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e. top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e. datatypes of the datasets
* Exploratory data analysis. mean, mediam, mode, count, min max etc.
* Droppimg the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Converting the continuous data into discrete form.
* Splitting the data into train and test datasets
* Initializing the DecisionTreeClassifier model and fit the training data into it and evaluate the model on test datasets and checking the accuracy and again evaluate the same model on train datasets and compare the accuracy.

In our calculation we found that test accuracy is 73% and the train accuracy is 100% so the model is overfit model

* To overcome the overfitting we use the RandomForestClassifier technique.
* Here we initialize the function by adding the different parameters like n\_features i.e no. of trees we want to make , n\_jobs, random\_state etc.and fit the training data into it and evaluate this model into test data and again on the training data.

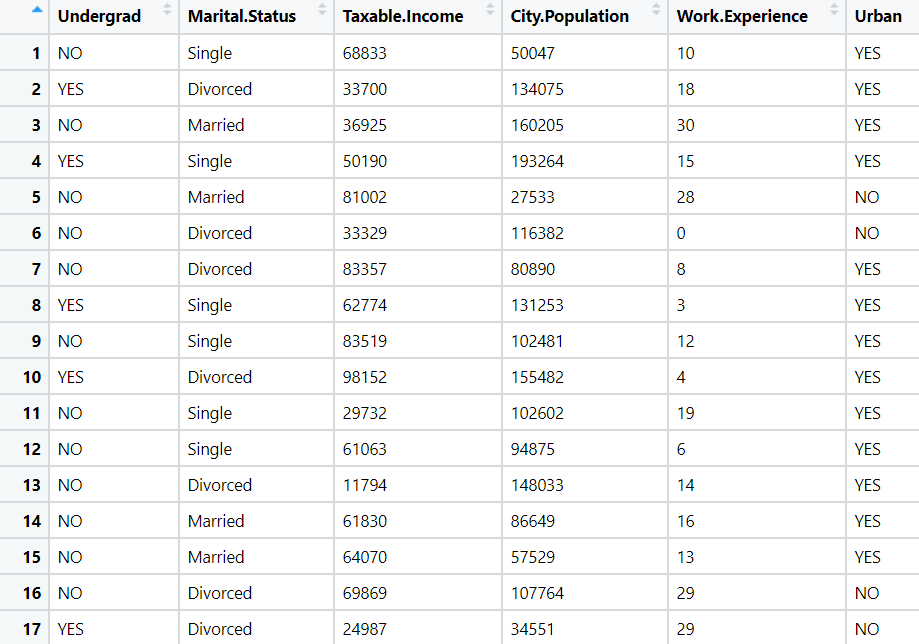
In our calculation we found that the test accuracy is 72% and train accuracy is 95% which is still overfit model but slightly better than previous model.

* Again we use GridSearchCv model to overcome the overfit model. Here we add the parameters Randoforestclassifier parameters and different hyperparameters like cv i.e cross validation, max\_features, min\_samples\_split to get the accuracy in the model and then initialize the GridSearchCV model and fit it into train datasets.
* We find best parameters and by taking this parameters we evaluate the model on the test datasets and again we evaluate the model on train datasets and compare the accuracy.

In our calculation we found that the test accuracy is 75% and train accuracy is 100% which is still overfit model but test accuracy is slightly greater than previous model.

1. Build a Decision Tree & Random Forest model on the fraud data. Treat those who have taxable\_income <= 30000 as Risky and others as Good (discretize the taxable

income column).





Ans: Business Objective :-

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| Undergrad | The customer is undergraduate or not | Qualitative, Nominal | Relevant |
| Marital. Status | Marital status of the customer | Qualitative, Nominal | Relevant |
| Taxable Income | Income of the customer | Quantitative, Ratio | Relevant |
| City Population | Population of the city where the customer lives | Quantitative, Ratio | Relevant |
| Work Experience | Work experience of the customer | Quantitative, Ratio | Relevant |
| Urban | Customer lives in Urban area or not | Qualitative, Nominal | Relevant |

To create the model to predict the new customer is good or risky to give loan or credit card.

* Import the libraries and different packages like pandas, numpy, matplotlib, sklearn.
* From sklearn library and tree package we import DecisionTreeClassifier function.It is used to create a classification model by building decision tree.
* From sklearn library and ensemble package import RandomForestClassifier function.
* 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.
* From sklearn library and model\_selction package import GridSearchCV function which is widely used for hyperparameter tuning to control the overfitting of the model.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e. top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e. datatypes of the datasets
* Exploratory data analysis. mean, mediam, mode, count, min max etc.
* Droppimg the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Converting the continuous data into discrete form.
* Splitting the data into train and test datasets
* Initializing the DecisionTreeClassifier model and fit the training data into it and evaluate the model on test datasets and checking the accuracy and again evaluate the same model on train datasets and compare the accuracy.

In our calculation we found that test accuracy is 57.5% and the train accuracy is 100% so the model is overfit model

* To overcome the overfitting we use the RandomForestClassifier technique.
* Here we initialize the function by adding the different parameters like n\_features i.e no. of trees we want to make , n\_jobs, random\_state etc.and fit the training data into it and evaluate this model into test data and again on the training data.

In our calculation we found that the test accuracy is 69% and train accuracy is 100% which is still overfit model .

* Again we use GridSearchCv model to overcome the overfit model. Here we add the parameters Randoforestclassifier parameters and different hyperparameters like cv i.e cross validation, max\_features, min\_samples\_split to get the accuracy in the model and then initialize the GridSearchCV model and fit it into train datasets.
* We find best parameters and by taking this parameters we evaluate the model on the test datasets and again we evaluate the model on train datasets and compare the accuracy.

In our calculation we found that the test accuracy is 74% and train accuracy is 82% which is still overfit model but test accuracy is slightly greater than previous model and the model is slightly better than previous model.

1. In the recruitment domain, HR faces the challenge of predicting if the candidate is faking their salary or not. For example, a candidate claims to have 5 years of experience and earns 70,000 per month working as a regional manager. The candidate expects more money than his previous CTC. We need a way to verify their claims (is 70,000 a month working as a regional manager with an experience of 5 years a genuine claim or does he/she make less than that?) Build a Decision Tree and Random Forest model with monthly income as the target variable.

A screenshot of a cell phone

Description automatically generated

Ans: Business Objective:-

To create a model to predict whether the salary asking by the employee is genuine claim or not.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Of Feature** | **Description** | **Type** | **Relevance** |
| Position of the employee | Position of the employee | Qualitative, ordinal | Relevant |
| no of Years of Experience of employee | No of experience the employee have | Quantitative, Ratio | Relevant |
| Monthly income of the employee | Monthly income of the employee | Quantitative, Ratio | Relevant |

* Import the libraries and different packages like pandas, numpy, matplotlib, sklearn.
* From sklearn library and tree package we import DecisionTreeClassifier function.It is used to create a classification model by building decision tree.
* From sklearn library and ensemble package import RandomForestClassifier function.
* 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.
* From sklearn library and model\_selction package import GridSearchCV function which is widely used for hyperparameter tuning to control the overfitting of the model.
* Loading the datasets and doing the univariate analysis and exploratory data analysis.
* Checking the head i.e. top 5 rows of the datasets
* Checking the columns names of the datasets
* Checking the null values if any available in dataset
* Checking the duplicate values in the datasets
* Checking the information i.e. datatypes of the datasets
* Exploratory data analysis. mean, mediam, mode, count, min max etc.
* Droppimg the unwanted column which is not useful for the analysis.
* Converting the nonnumerical data into numerical data by using one hot encoding or Label Encoder or pandas get\_dummies function as per the requirement
* Converting the continuous data into discrete form.
* Splitting the data into train and test datasets
* Initializing the DecisionTreeClassifier model and fit the training data into it and evaluate the model on test datasets and checking the accuracy and again evaluate the same model on train datasets and compare the accuracy.

In our calculation we found that test accuracy is 95% and the train accuracy is 99% so the model is overfit model

* To overcome the overfitting we use the RandomForestClassifier technique.
* Here we initialize the function by adding the different parameters like n\_features i.e no. of trees we want to make , n\_jobs, random\_state etc.and fit the training data into it and evaluate this model into test data and again on the training data.

In our calculation we found that the test accuracy is 95% and train accuracy is 87% which is still underfit model .

* Again we use GridSearchCv model to overcome the underfit model. Here we add the parameters Randoforestclassifier parameters and different hyperparameters like cv i.e cross validation, max\_features, min\_samples\_split to get the accuracy in the model and then initialize the GridSearchCV model and fit it into train datasets.
* We find best parameters and by taking this parameters we evaluate the model on the test datasets and again we evaluate the model on train datasets and compare the accuracy.

In our calculation we found that the test accuracy is 90% and train accuracy is 98% which is still overfit model but the model is slightly better than previous model.