CS 584-04: Machine Learning

Autumn 2019 Assignment 3

You are asked to use a decision tree model to predict the usage of a car. The data is the claim_history.csv which has 10,302 observations. The analysis specifications are:

Target Variable

- **CAR_USE**. The usage of a car. This variable has two categories which are *Commercial* and *Private*. The *Commercial* category is the Event value.

Nominal Predictor

- **CAR_TYPE**. The type of a car. This variable has six categories which are *Minivan*, *Panel Truck*, *Pickup*, *SUV*, *Sports Car*, and *Van*.
- **OCCUPATION**. The occupation of the car owner. This variable has nine categories which are *Blue Collar*, *Clerical, Doctor*, *Home Maker*, *Lawyer*, *Manager*, *Professional*, *Student*, and *Unknown*.

Ordinal Predictor

- **EDUCATION**. The education level of the car owner. This variable has five ordered categories which are *Below High School* < *High School* < *Bachelors* < *Masters* < *Doctors*.

Analysis Specifications

- **Partition**. Specify the target variable as the stratum variable. Use stratified simple random sampling to put 70% of the records into the Training partition, and the remaining 30% of the records into the Test partition. The random state is 27513.
- **Decision Tree**. The maximum number of branches is two. The maximum depth is two. The split criterion is the Entropy metric.

You need to write a few Python programs to assist you in answering the questions.

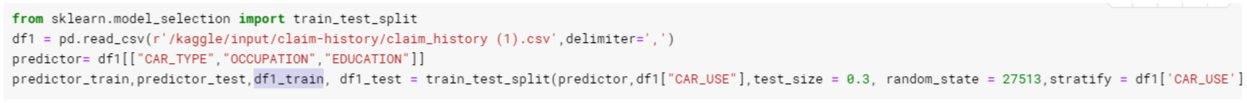
Question 1 (20 points)

Please provide information about your Data Partition step.

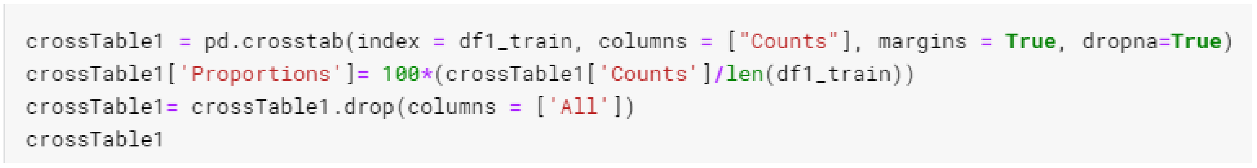
a) (5 points). Please provide the frequency table (i.e., counts and proportions) of the target variable in the Training partition?

Ans:

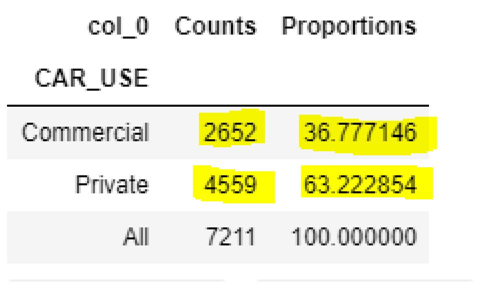
As per given information I have partitioned 70% of data records into training partition. To find frequency table I have used crosstab function of pandas which Compute a simple cross-tabulation of two (or more) factors. In df1_train I have saved only training data's CAR_USE data. So, I am counting commercial and private CAR_USE and their proportions with respect to training data and forming frequency table as follows,



```
from sklearn.model_selection import train_test_split
df1 = pd.read_csv(r'/kaggle/input/claim-history/claim_history (1).csv',delimiter=',')
predictor= df1[["CAR_TYPE","OCCUPATION","EDUCATION"]]
predictor_train,predictor_test,df1_train, df1_test = train_test_split(predictor,df1["CAR_USE"],test_size = 0.3, random_state = 27513,stratify = df1['CAR_USE']
```



```
crossTable1 = pd.crosstab(index = df1_train, columns = ["Counts"], margins = True, dropna=True)
crossTable1['Proportions']= 100*(crossTable1['Counts']/len(df1_train))
crossTable1= crossTable1.drop(columns = ['All'])
crossTable1
```

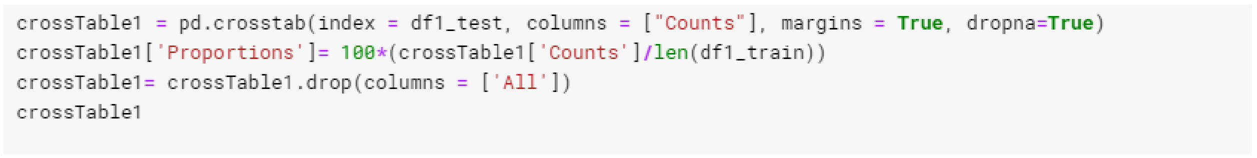


col_0	Counts	Proportions
CAR_USE		
Commercial	2652	36.777146
Private	4559	63.222854
All	7211	100.000000

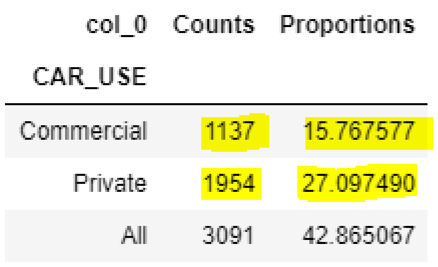
b) (5 points). Please provide the frequency table (i.e., counts and proportions) of the target variable in the Test partition?

Ans:

As per given information I have partitioned 30% of data records into testing partition. To find frequency table I have used crosstab function of pandas which Compute a simple cross-tabulation of two (or more) factors. In df1_test I have saved only testing data's CAR_USE data. So, I am counting commercial and private CAR_USE and their proportions with respect to testing data and forming frequency table as follows,



```
crossTable1 = pd.crosstab(index = df1_test, columns = ["Counts"], margins = True, dropna=True)
crossTable1['Proportions']= 100*(crossTable1['Counts']/len(df1_train))
crossTable1= crossTable1.drop(columns = ['All'])
crossTable1
```



col_0	Counts	Proportions
CAR_USE		
Commercial	1137	15.767577
Private	1954	27.097490
All	3091	42.865067

c) (5 points). What is the probability that an observation is in the Training partition given that CAR_USE = *Commercial*?

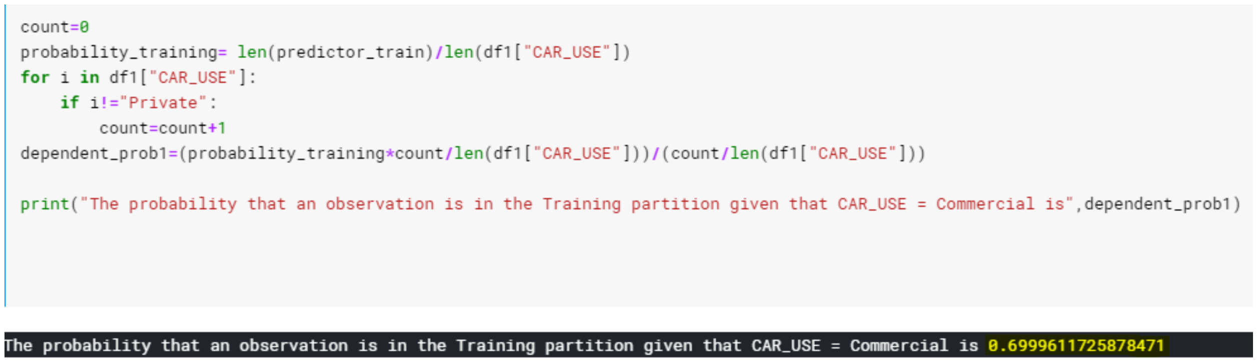
Ans:

Calculating probability that observation is training partition given that CAR_USE= Commercial is conditional probability. To calculate conditional probability, we us Bayes' theorem of probability is,

$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$

Here in our example,

$P(Training \mid Commercial=1) = \frac{P(Commercial=1|Training)P(Training)}{P(Commercial=1)}$ = 0.6999611725878471



```
count=0
probability_training= len(predictor_train)/len(df1["CAR_USE"])
for i in df1["CAR_USE"]:
    if i!="Private":
        count=count+1
dependent_prob1=(probability_training*count/len(df1["CAR_USE"]))/(count/len(df1["CAR_USE"]))

print("The probability that an observation is in the Training partition given that CAR_USE = Commercial is",dependent_prob1)
```

```
The probability that an observation is in the Training partition given that CAR_USE = Commercial is 0.6999611725878471
```

d) (5 points). What is the probability that an observation is in the Test partition given that CAR_USE = *Private*?

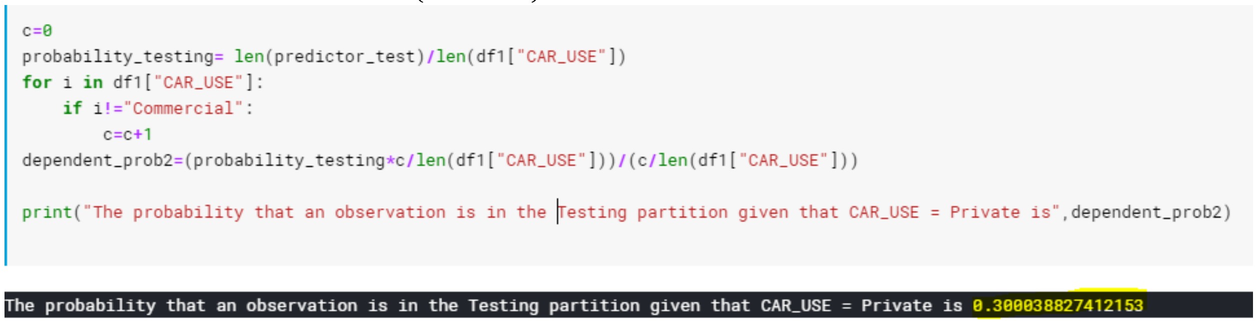
Ans:

Calculating probability that observation is training partition given that CAR_USE= Commercial is conditional probability. To calculate conditional probability, we us Bayes' theorem of probability is,

P(A|B)=$\frac{P(B|A)P(A)}{P(B)}$

Here in our example,

P(Testing | Private=1) = $\frac{P(Private=1|Testing)P(Testing)}{P(Private=1)}$ = 0.300038827412153



```
c=0
probability_testing= len(predictor_test)/len(df1["CAR_USE"])
for i in df1["CAR_USE"]:
    if i!="Commercial":
        c=c+1
dependent_prob2=(probability_testing*c/len(df1["CAR_USE"]))/(c/len(df1["CAR_USE"]))

print("The probability that an observation is in the Testing partition given that CAR_USE = Private is",dependent_prob2)
```

```
The probability that an observation is in the Testing partition given that CAR_USE = Private is 0.300038827412153
```

Question 2 (40 points)

Please provide information about your decision tree.

a) (5 points). What is the entropy value of the root node?

Ans:

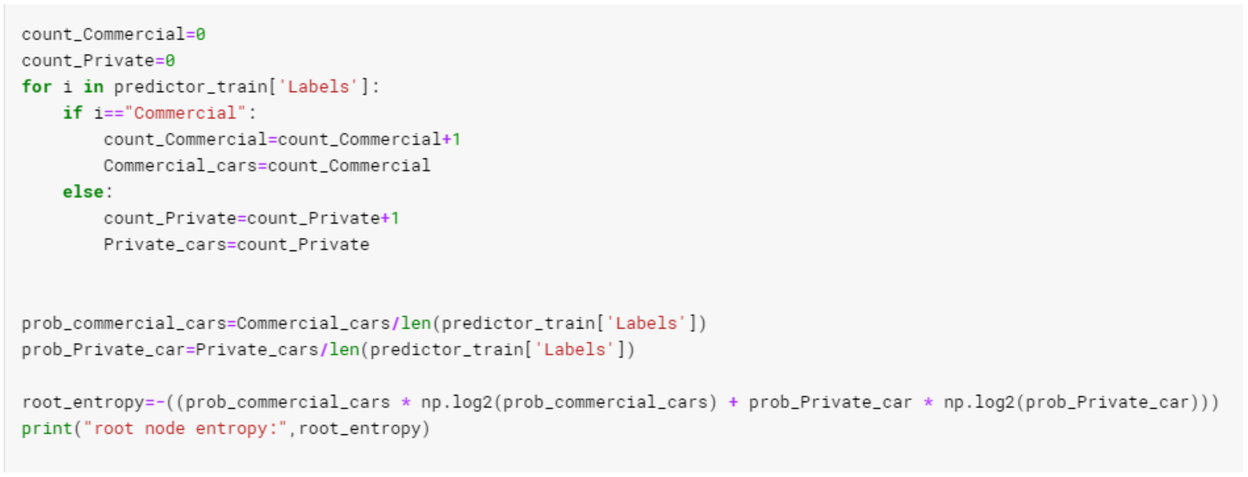
Entropy (T) = $\sum \mathbf{p(j | t) \log p(j | t)}$

Entropy (P): parent node entropy which is split into k partitions. It is a measure of impurity of split.

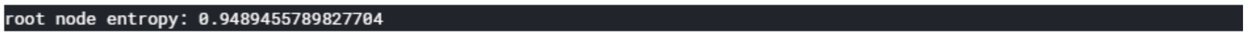
p(j | t) is the relative frequency of class j at node t.

To calculate root node entropy, we will consider total count of each class i.e. Commercial and Private with respect to training data.

Root node entropy = 0.9489455789827704



```
count_Commercial=0
count_Private=0
for i in predictor_train['Labels']:
    if i=="Commercial":
        count_Commercial=count_Commercial+1
        Commercial_cars=count_Commercial
    else:
        count_Private=count_Private+1
        Private_cars=count_Private


prob_commercial_cars=Commercial_cars/len(predictor_train['Labels'])
prob_Private_car=Private_cars/len(predictor_train['Labels'])

root_entropy=-((prob_commercial_cars * np.log2(prob_commercial_cars) + prob_Private_car * np.log2(prob_Private_car)))
print("root node entropy:",root_entropy)
```



```
root node entropy: 0.9489455789827704
```

b) (5 points). What is the split criterion (i.e., predictor name and values in the two branches) of the first layer?

Ans:

As discussed in class number of splits depends on the number of distinct values of predictor in parent node. If predictor has n different values in parent the number of possible splits is as follows,

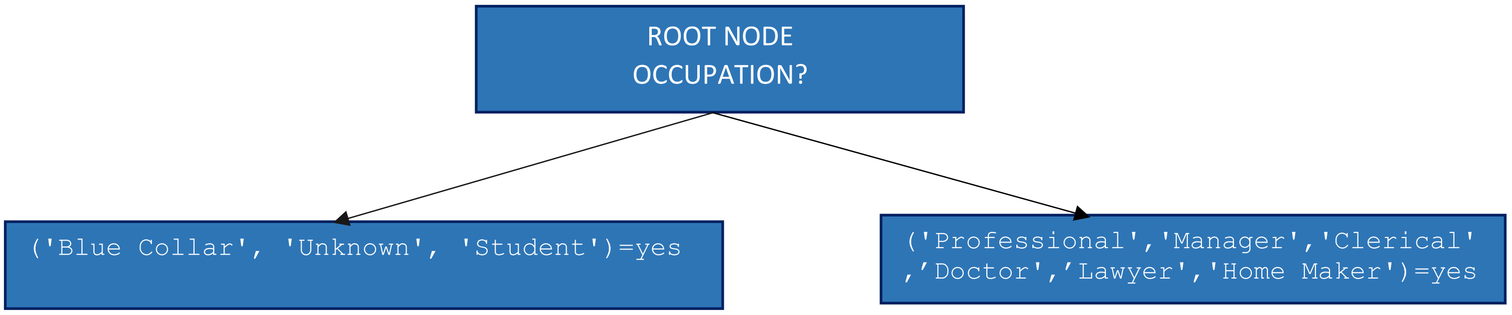
For Nominal,

2^{n-1}-1 therefore in given problem for CAR_TYPE(31)and OCCUPATION(255) we will calculate splits as this.

For Ordinal,

n-1 in given example we will calculate splits of EDUCATION(5)as this.

OCCUPATION is the 1st split predictor as the entropy value of OCCUPATION is smallest amongst all attributes. Values in two branches are as follows,

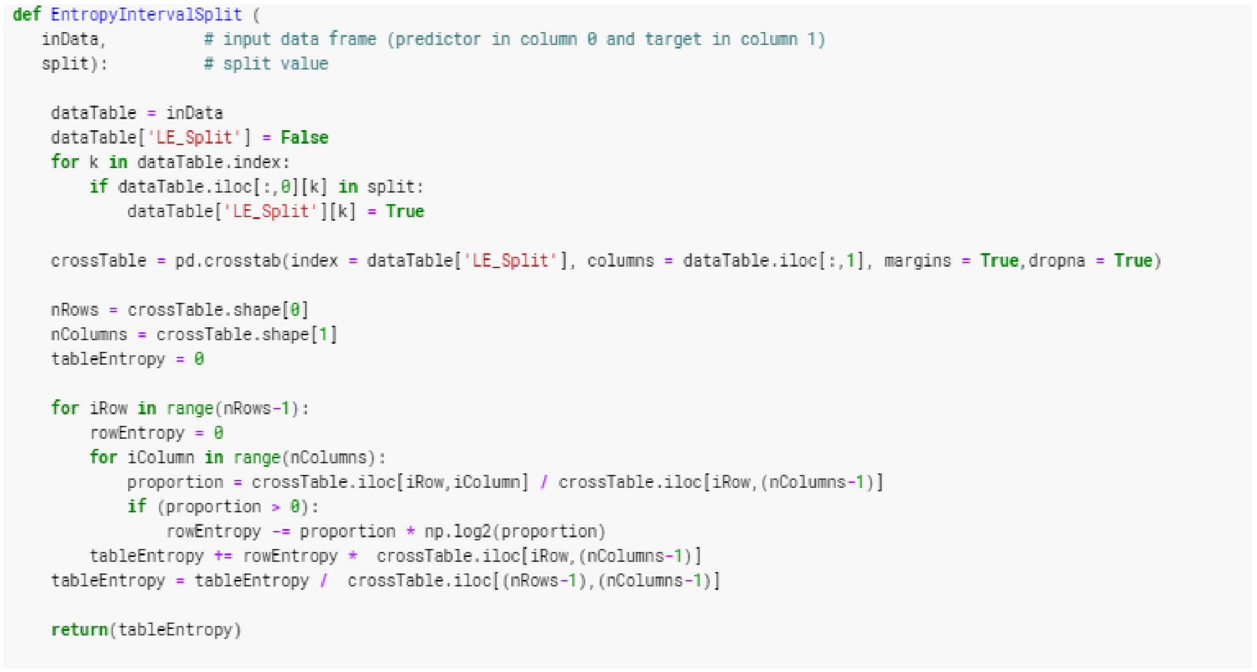

ROOT NODE
OCCUPATION?
('Blue Collar', 'Unknown', 'Student')=yes
('Professional','Manager','Clerical','Doctor','Lawyer','Home Maker')=yes

To find 1st split(best split):

- Calculate entropy values of each attribute as the impurity measure used determining goodness of split is entropy.(CAR_TYPE,OCCUPATION,EDUCATION)
- Select attribute with lowest entropy value as, the distribution is clearer for low entropy value we will get more information from such node. (OCCUPATION(0.7148805225259208)<CAR_TYPE(0.7573352263531922)<EDUCATION(0.9343298080392602))
- Hence, OCCUPATION is chosen over other attributes.

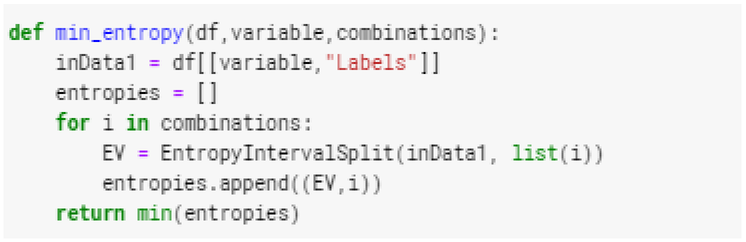
c) (10 points). What is the entropy of the split of the first layer?

Entropy value of 1st layer is: 0.7148805225259208



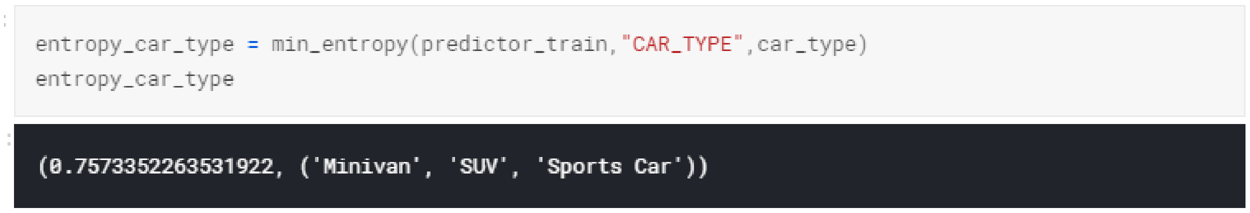
```python
def EntropyIntervalSplit (
   inData,          # input data frame (predictor in column 0 and target in column 1)
   split):          # split value

    dataTable = inData
    dataTable['LE_Split'] = False
    for k in dataTable.index:
        if dataTable.iloc[:,0][k] in split:
            dataTable['LE_Split'][k] = True

    crossTable = pd.crosstab(index = dataTable['LE_Split'], columns = dataTable.iloc[:,1], margins = True,dropna = True)

    nRows = crossTable.shape[0]
    nColumns = crossTable.shape[1]
    tableEntropy = 0

    for iRow in range(nRows-1):
        rowEntropy = 0
        for iColumn in range(nColumns):
            proportion = crossTable.iloc[iRow,iColumn] / crossTable.iloc[iRow,(nColumns-1)]
            if (proportion > 0):
                rowEntropy -= proportion * np.log2(proportion)
        tableEntropy += rowEntropy *  crossTable.iloc[iRow,(nColumns-1)]
    tableEntropy = tableEntropy /  crossTable.iloc[(nRows-1),(nColumns-1)]

    return(tableEntropy)
```

Below is the user defined function to find minimum entropy,



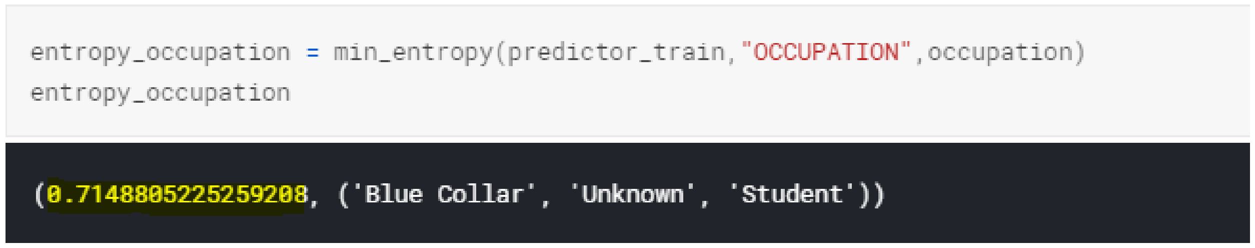
```python
def min_entropy(df,variable,combinations):
    inData1 = df[[variable,"Labels"]]
    entropies = []
    for i in combinations:
        EV = EntropyIntervalSplit(inData1, list(i))
        entropies.append((EV,i))
    return min(entropies)
```

Calling above function and passing respective dataframes for CAR_TYPE, OCCUPATION and EDUCATION with respect to train data.



```python
entropy_car_type = min_entropy(predictor_train,"CAR_TYPE",car_type)
entropy_car_type
```

```
(0.7573352263531922, ('Minivan', 'SUV', 'Sports Car'))
```



```python
entropy_occupation = min_entropy(predictor_train,"OCCUPATION",occupation)
entropy_occupation
```

```
(0.7148805225259208, ('Blue Collar', 'Unknown', 'Student'))
```

```
entropy_edu = min_entropy(predictor_train,"EDUCATION",education)
entropy_edu
```

```
(0.9343298080392602, ('Below High School',))
```

Here we are using entropy as a measure to find goodness of split. Lowest value of entropy is preferred over higher value. Hence 1st split will be at OCCUPATION with entropy value of 0.7148805225259208

d) (5 points). How many leaves?

Ans: The number of leaves in decision tree can be calculated as 2^n where n is the max depth.

In our problem max depth=2 hence number of leaves=2^2=**4**

e) (15 points). Describe all your leaves. Please include the decision rules and the counts of the target values.
Ans: From the given problem decision tree formed is having max depth of 2 and 4 leaf nodes. The tree formed is as follows,

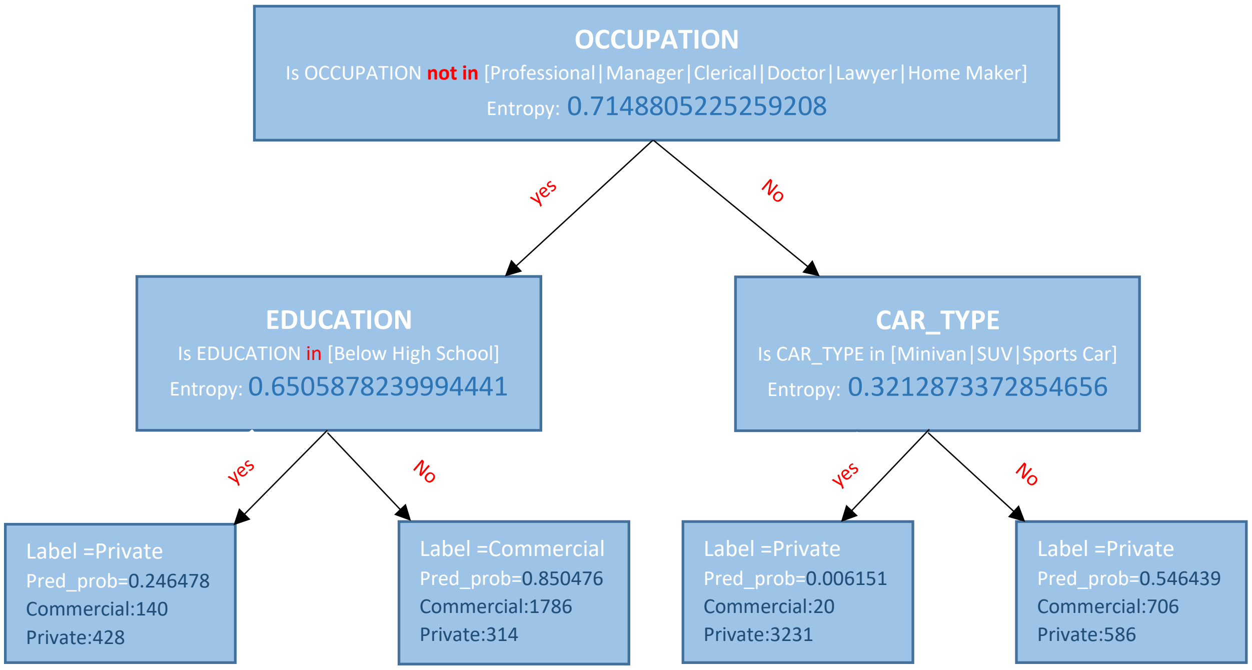
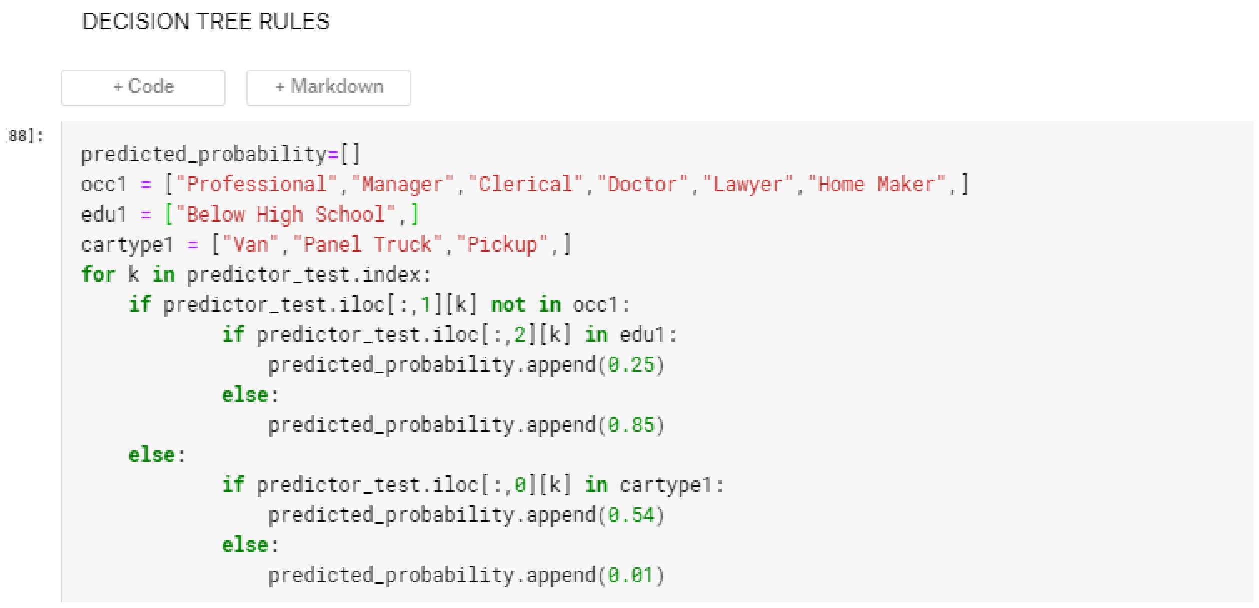


Diagram above shows traversal of our decisions from root node till leaf nodes to predict target values for test observation. As decision tree is greedy approach, at every node we use entropy as a splitting criterion to measure goodness of a node and selects attributes from OCCUPATION, EDUCATION AND CAR_TYPE.
In diagram above, Pred_prob is predicted probability of an event. In given problem we have given event as Commercial. Hence compared these predicted probabilities with threshold

value(probability of commercial cars) and if predicted probability is greater than or equal to threshold then it is labeled as Commercial.
DECISION RULES are as follows,



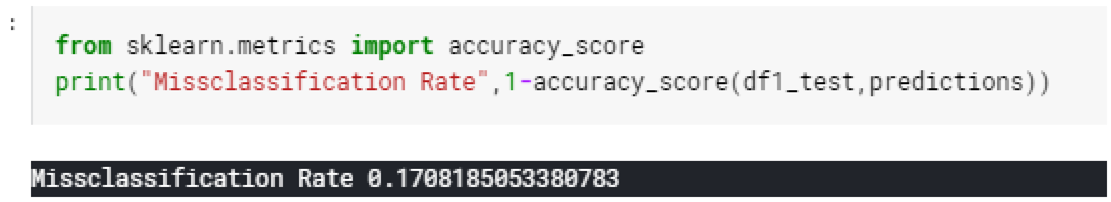
```
88]:
predicted_probability=[]
occ1 = ["Professional","Manager","Clerical","Doctor","Lawyer","Home Maker",]
edu1 = ["Below High School",]
cartype1 = ["Van","Panel Truck","Pickup",]
for k in predictor_test.index:
    if predictor_test.iloc[:,1][k] not in occ1:
            if predictor_test.iloc[:,2][k] in edu1:
                predicted_probability.append(0.25)
            else:
                predicted_probability.append(0.85)
    else:
            if predictor_test.iloc[:,0][k] in cartype1:
                predicted_probability.append(0.54)
            else:
                predicted_probability.append(0.01)
```

Question 3 (40 points)

Please apply your decision tree to the Test partition and then provide the following information.

a) (10 points). Use the proportion of target Event value in the training partition as the threshold, what is the Misclassification Rate in the Test partition?

Ans: Misclassification Rate in the Test partition: **0.1708185053380783**



```
from sklearn.metrics import accuracy_score
print("Missclassification Rate",1-accuracy_score(df1_test,predictions))
```

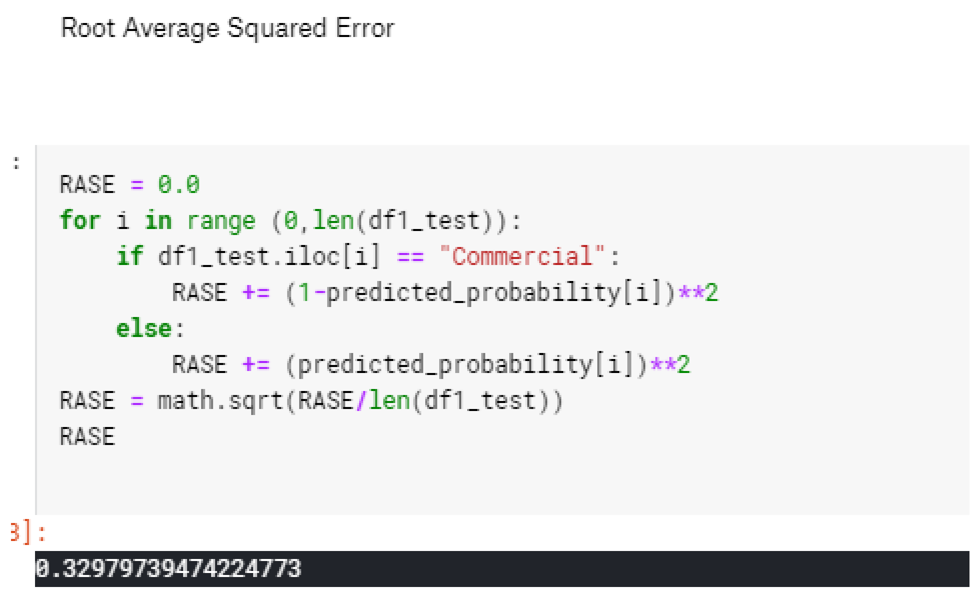
```
Missclassification Rate 0.1708185053380783
```

Missclassification rate = $\frac{n(\boldsymbol{Missclassified\ observations})}{n(\boldsymbol{Observations})}$

So, we can say observation is misclassified if the observed event is Commercial but predicted event is non-event that is Private.

b) (10 points). What is the Root Average Squared Error in the Test partition?

Ans:
The Root Average Squared Error in the Test partition: **0.32979739474224773**



Root Average Squared Error

```
:
RASE = 0.0
for i in range (0,len(df1_test)):
    if df1_test.iloc[i] == "Commercial":
        RASE += (1-predicted_probability[i])**2
    else:
        RASE += (predicted_probability[i])**2
RASE = math.sqrt(RASE/len(df1_test))
RASE
```

```
3]:
0.32979739474224773
```

If model fits data well then, we consider probabilities to be:
Closer to 1 if it is an event (Commercial)
Closer to 0 if it is non-event (Private)
Hence, Root Average Squared Error is a metric that shows how close, on the average, the predicted event probabilities are to their respective anticipated values.

(Average Squared Error) ASE = $\frac{\sum_{l=1}^{n_E}(1-p_{l1})^2+\sum_{k=1}^{n_{NE}}(0-p_{k0})^2}{n_E+n_{NE}}$

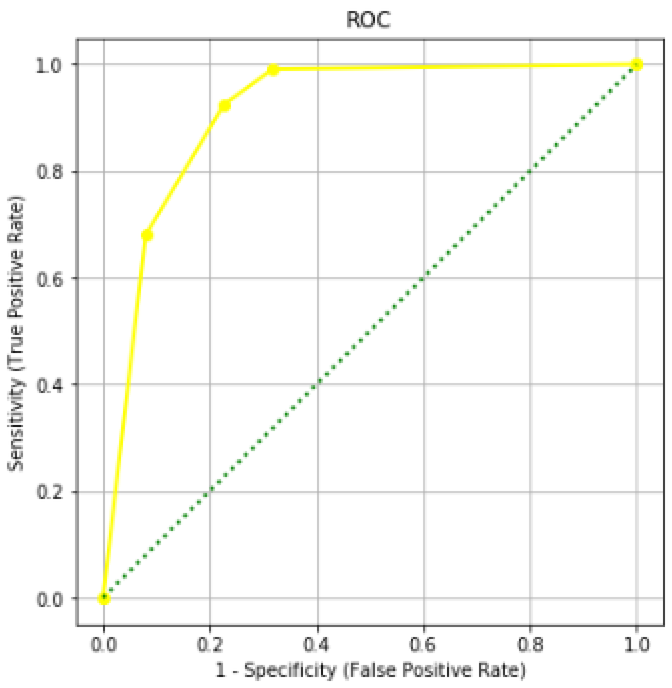
Root Average Squared Error=$\sqrt{Average\ Squared\ Error}$
Smaller the RASE that is below 0.5, better is the model. We are getting RASE below 0.5 hence our model is **better model**.

c) (10 points). What is the Area Under Curve in the Test partition?

Ans:
Area Under Curve in the Test partition: **0.9114708659772841**
Area Under Curve is a metric that measures how close the model to the ideal scenario or how far away from the worst scenario. To get the area under the curve for our decision model, we use the predicted observations and actual observations and check how the model can correctly classify the observations. Higher the AUC better is the model(>0.5) We have **Better Model** as AUC>0.5.

d) (10 points). Generate the Receiver Operating Characteristic curve for the Test partition. The axes must be properly labeled. Also, don't forget the diagonal reference line.

Ans:
ROC Curve is represented by FPR on x-axis and TPR on y-axis. Curve very close to diagonal is worst model while curve located on upper left corner is a good classification. Below is ROC curve for given example,



From graph above green dotted line is a diagonal and yellow lined curve is ROC curve which is in upper left corner of the graph hence we can say that model is ideal model.