Statistical Machine Learning 2018

Coursework 1 Report

Part I - Statistics

Question 1:

Features	Types of Features
Age	Numeric
Workclass	Categorical
Final Weight	Numeric
Education	Categorical
Education-num	Categorical
Marital Status	Categorical
Occupation	Categorical
Relationship	Categorical
Race	Categorical
Sex	Categorical
Capital Gain	Numeric
Capital Loss	Numeric
Hours per week	Numeric
Native Country	Categorical
Income	Binary

The Edu-Num columns in the all three data sets are ignored, as the information in the column is just the numeric value for the Education column. It is sufficient to just investigate the Education column. The learning and testing process are done without considering the Edu-Num column for all sets of data.

Question 2:

(a) Size of original training daa set = 29999
Size of training data set after removing samples missing values = 27776

(b) Size of class (Income <= 50k) = 20837
Size of class (Income >50k) = 6939
Frequency table for each predictor:

Features (Workclass)	Frequency
Federal-gov	876
Local-gov	1894
Private	20497
Self-emp-inc	999
Self-emp-not-inc	2314
State-gov	1183
Without-pay	13

Features (Marital Status)	Frequency
Divorced	3857
Married-AF-spouse	21
Married-civ-spouse	12980
Married-spouse-absent	335
Never-married	8976
Separated	858
Widowed	749

Features (Native Country)	Frequency
Cambodia	16
Canada	99
China	59
Columbia	53
Cuba	85
Dominican-Republic	61
Ecuador	24
El-Salvador	92
England	81
France	26
Germany	119
Greece	28
Guatemala	56
Haiti	34
Holand-Netherlands	1
Honduras	12
Hong	16
Hungary	11
India	89
Iran	36
Ireland	24
Italy	60
Jamaica	73
Japan	52
Laos	14
Mexico	551
Nicaragua	32
Outlying-US(Guam-USVI-etc)	14
Peru	28
Philippines	175
Poland	46
Portugal	30
Puerto-Rico	99
Scotland	9
South	65
Taiwan	42
Thailand	16
Trinadad&Tobago	18
United-States	25359
Vietnam	56
Yugoslavia	15

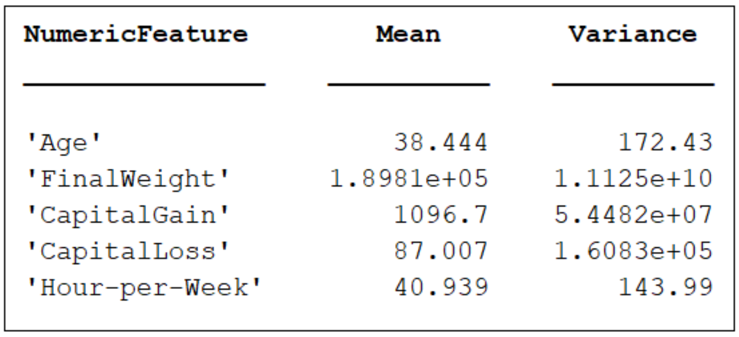
Features (Relationship)	Frequency
Adm-clerical	3404
Armed-Forces	9
Craft-repair	3735
Exec-managerial	3698
Farming-fishing	905
Handlers-cleaners	1238
Machine-op-inspct	1819
Other-service	2939
Priv-house-serv	122
Prof-specialty	3738
Protective-serv	591
Sales	3282
Tech-support	833
Transport-moving	1463
Husband	11508
Not-in-family	7108
Other-relative	829
Own-child	4097
Unmarried	2940
Wife	1294

Features (Sex)	Frequency
Female	8975
Male	18801

Features (Race)	Frequency
Amer-Indian-Eskimo	262
Asian-Pac-Islander	818
Black	2567
Other	215
White	23914

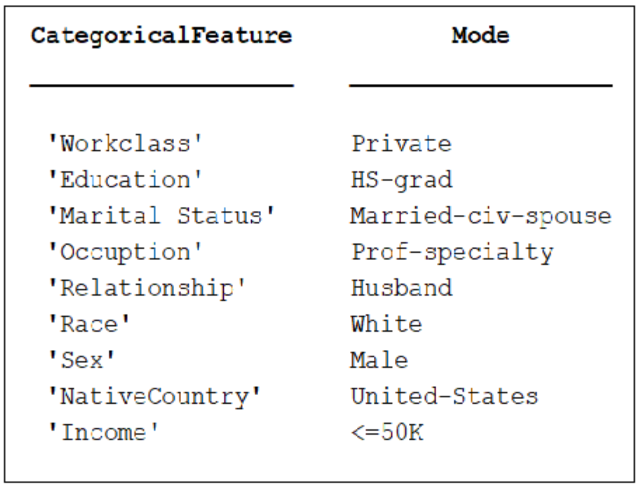
Features (Education)	Frequency
10th	760
11th	976
12th	341
1st-4th	132
5th-6th	264
7th-8th	516
9th	417
Assoc-acdm	926
Assoc-voc	1211
Bachelors	4659
Doctorate	347
HS-grad	9060
Masters	1511
Preschool	44
Prof-school	502
Some-college	6110

(c) Numerical Features:



```
NumericFeature          Mean         Variance
________________     __________     __________

'Age'                    38.444         172.43
'FinalWeight'        1.8981e+05     1.1125e+10
'CapitalGain'            1096.7     5.4482e+07
'CapitalLoss'            87.007     1.6083e+05
'Hour-per-Week'          40.939         143.99
```

Categorical Features:

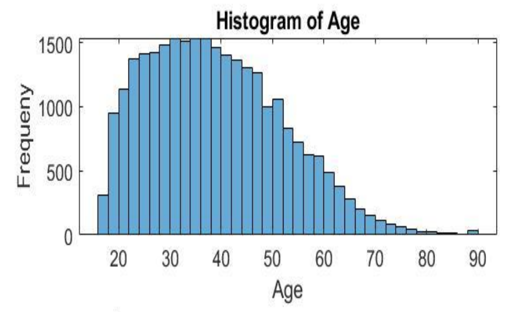


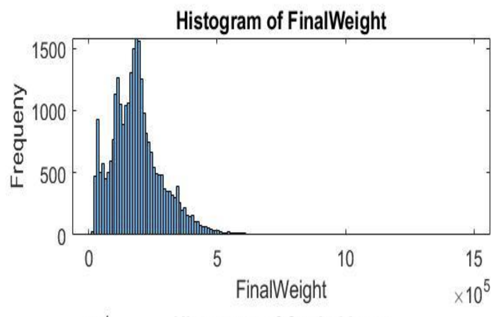
```
CategoricalFeature          Mode
__________________    __________________

 'Workclass'          Private
 'Education'          HS-grad
 'Marital Status'     Married-civ-spouse
 'Occuption'          Prof-specialty
 'Relationship'       Husband
 'Race'               White
 'Sex'                Male
 'NativeCountry'      United-States
 'Income'             <=50K
```

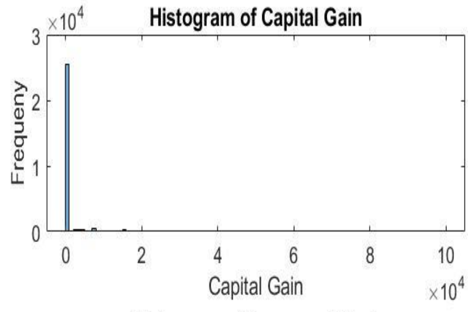
(d) The statistics result showed that there are 20837 samples with the income of <=50K in the training sets and 6939 samples with income of >50K. The ratio of distribution between these two labels is 0.3330, this showed an obvious skewness on the distribution. So, the data set is said to be unbalanced.

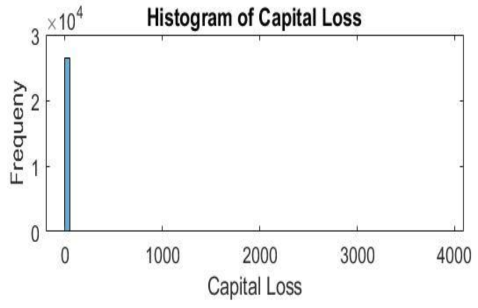
Question 3:

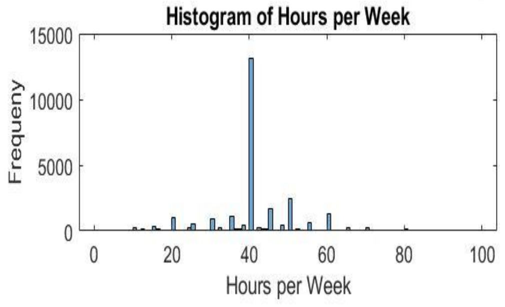
Histograms for Numerical Features



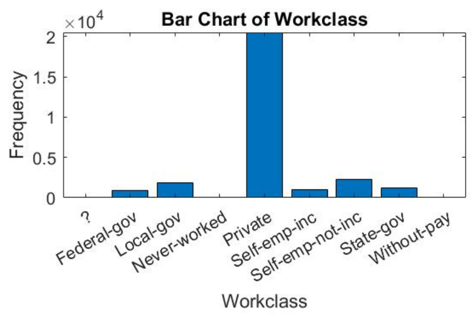


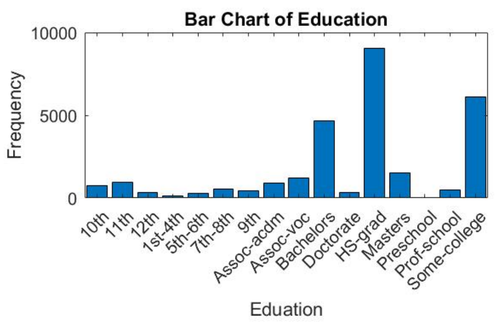


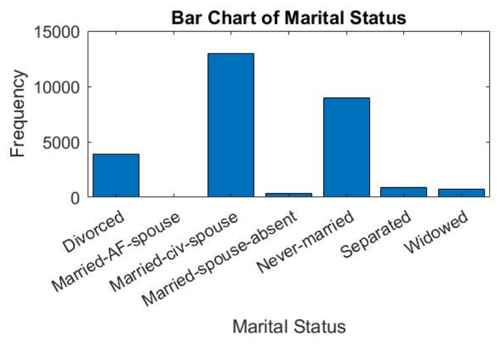


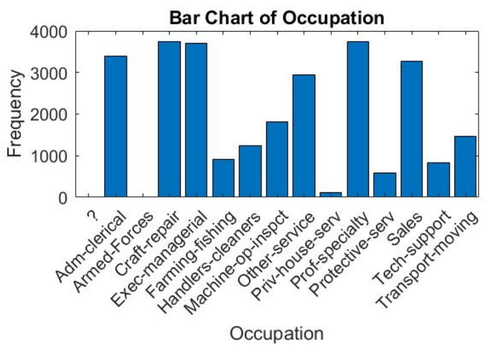


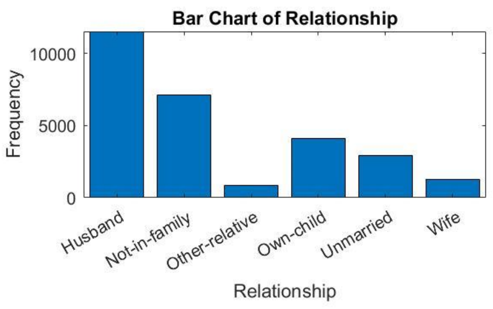
Bar chart for Categorical Features

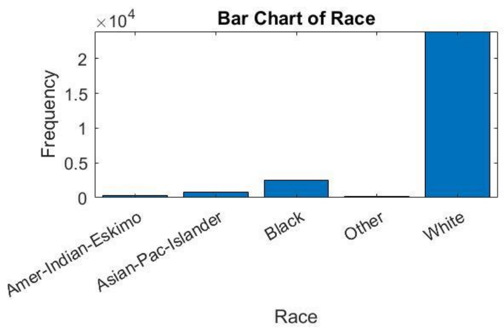


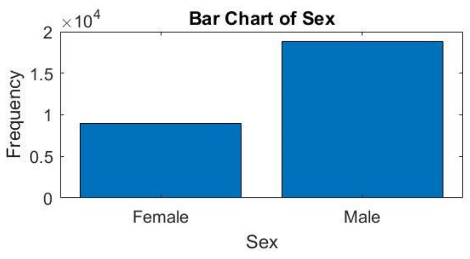


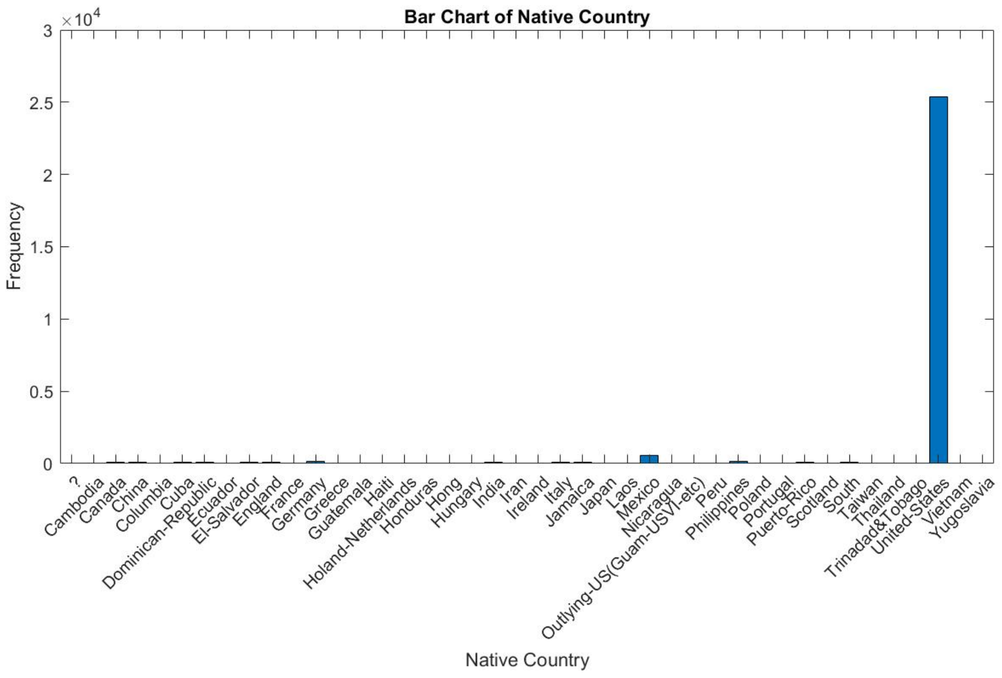





Bar Chart of Relationship
Frequency
10000
5000
0
Husband
Not-in-family
Other-relative
Own-child
Unmarried
Wife
Relationship


Bar Chart of Race
×10^4
Frequency
2
1.5
1
0.5
0
Amer-Indian-Eskimo
Asian-Pac-Islander
Black
Other
White
Race

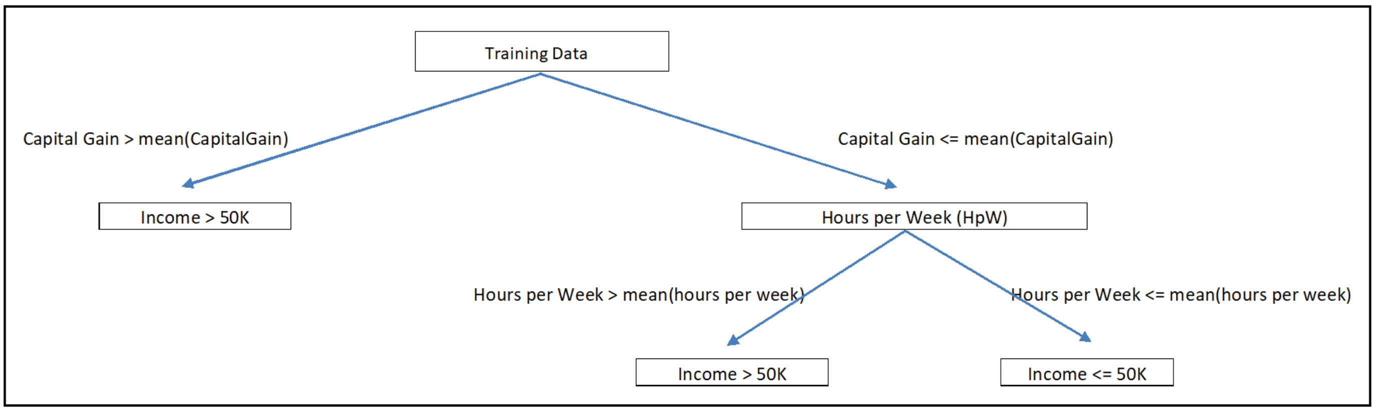

Bar Chart of Sex
×10^4
Frequency
2
1.5
1
0.5
0
Female
Male
Sex


Bar Chart of Native Country
×10^4
Frequency
3
2.5
2
1.5
1
0.5
0
?
Cambodia
Canada
China
Columbia
Cuba
Dominican-Republic
Ecuador
El-Salvador
England
France
Germany
Greece
Guatemala
Haiti
Holand-Netherlands
Honduras
Hong
Hungary
India
Iran
Ireland
Italy
Jamaica
Japan
Laos
Mexico
Nicaragua
Outlying-US(Guam-USVI-etc)
Peru
Philippines
Poland
Portugal
Puerto-Rico
Scotland
South
Taiwan
Thailand
Trinadad&Tobago
United-States
Vietnam
Yugoslavia
Native Country

Part II – Decision Tree

Question 1:

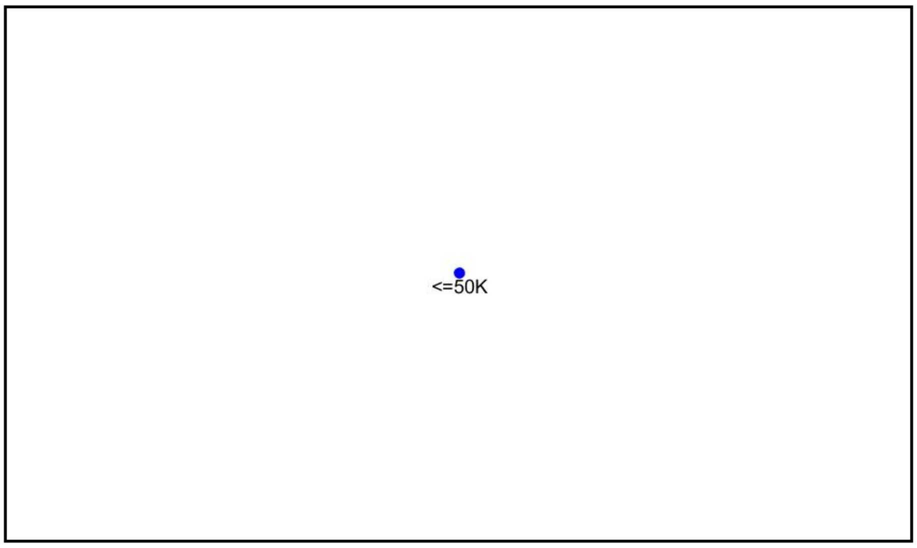
The features selected are Capital Gain and Hours per Week. The decision tree constructed is


Training Data
Capital Gain > mean(CapitalGain)
Capital Gain <= mean(CapitalGain)
Income > 50K
Hours per Week (HpW)
Hours per Week > mean(hours per week)
Hours per Week <= mean(hours per week)
Income > 50K
Income <= 50K

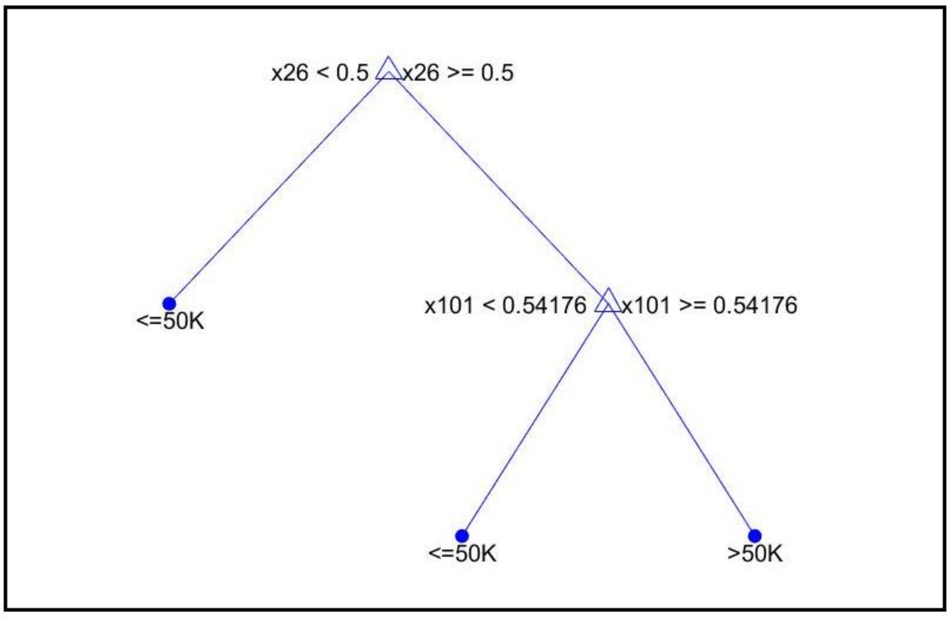
Question 2:

For Training Dataset

(a) Maximum number of splits = 1
Training Error = 0.2498 = 24.98%
Accuracy = 0.7502 = 75.02%
Decision Tree:



(b) Maximum number of splits = 2
Training Error = 0.2121 = 21.21%
Accuracy = 0.7879 = 78.79%
Decision Tree:

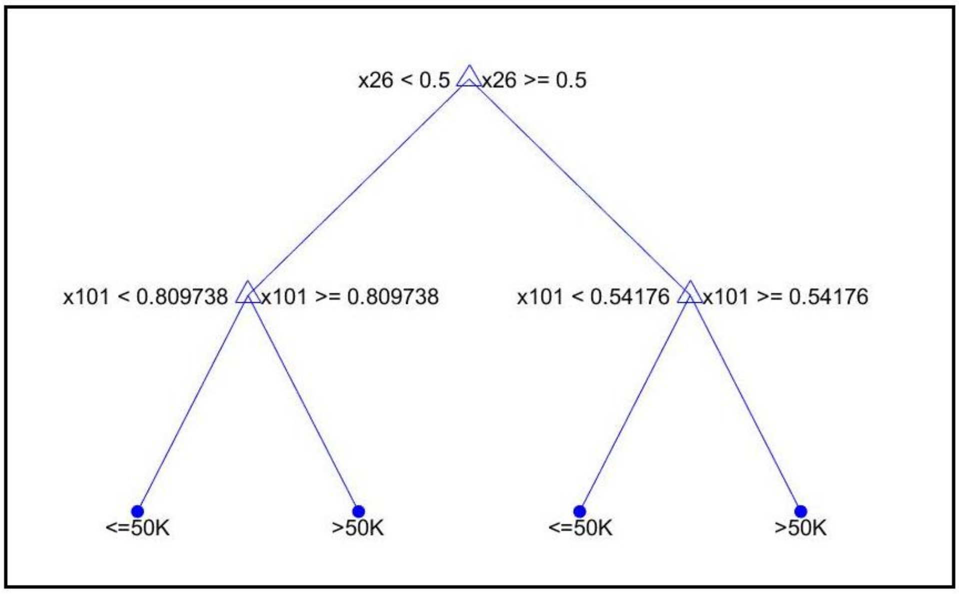


Maximum number of splits = 3

Training Error = 0.2027 = 20.27%

Accuracy = 0.7973 = 79.73%

Decision Tree:



The decision tree with maximum splits of 1 has the largest error of 24.98%, as the tree simply classified the samples input as negative result, which is those income with <=50K. As the decision tree progress further, the error rate decreases. The degree of decrement in error rate is getting smaller from splits of 2 to 3.

(c)

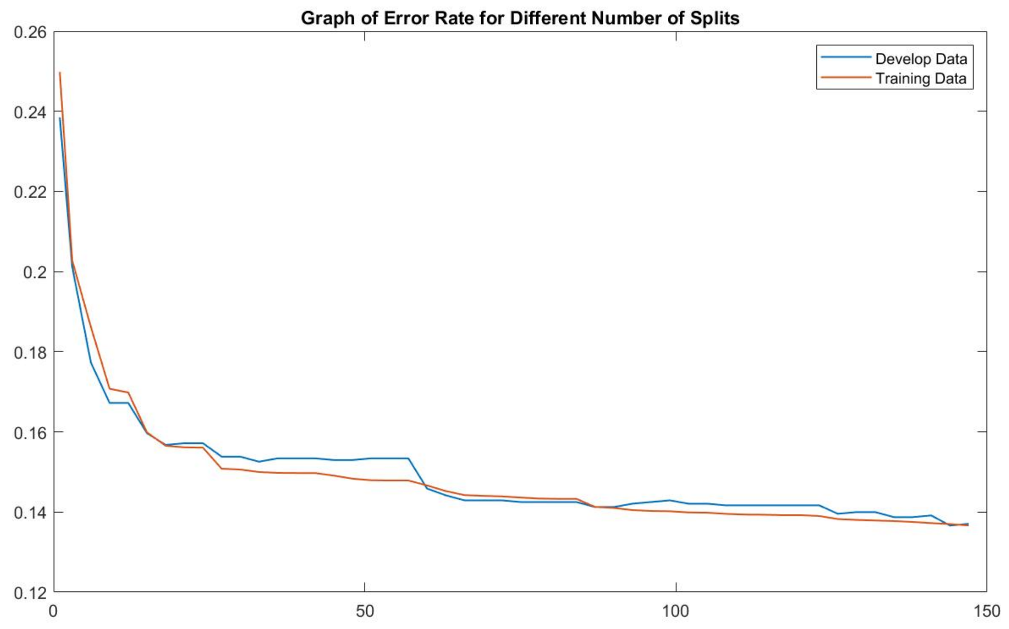
	Loss	
Splits	**Training Data**	**Develop Data**
1	0.24982	0.24982
2	0.21213	0.21777
3	0.20273	0.21069

As the number of splits increase from 1 to 3, the error rate on both training data and the development data are decreasing. As the number of splits is 1, both sets of data have the same error rate of 24.98%. However, as the split increases from 2 to 3, the error on the develop data is higher than the training data. As the depth increases, more features are taken to build the tree, with the present of more important information which can be influential, better classification can be done, and lower the error rate.

Question 3:

Tuning hyperparameter of the model trained.

Plot of the result:



Decision:

From the result of error rate computed on the development data based on the decision tree trained for a range of different number of splits (from 1 to 147), error rate has been relatively constant with slight fluctuation along the end of the graph. The tree has the minimum error of 13.663% with 144 splits. According to the Occam Learning, we choose the simplest tree for the same error rate. Therefore, the chosen model is the decision tree with 144 splits.

Question 4:

The chosen model is decision tree with 144 splits.

Test error rate = 0.1465 = 14.65%

Accuracy = 85.35%

Part III – K-Nearest Neighbours

Question 1

For integer encode, [0,1,…,N], the categorical feature with a wide range of unique entries, the categorical predictors may take up to a very large value, N. This will be an issue when we are trying to build other models with this type of representation. Furthermore, K-nearest neighbour is the classifier that compute the distance between the query sample and the other existing examples in the model. When we use integer encode, the distance computed will be of a wide range for the feature. By applying the one hot encoding, the distance between all unique entries for one feature will be the same. This reduces the complexity of the computation and classification process. Hence, one hot encoding is a better idea of data representation.

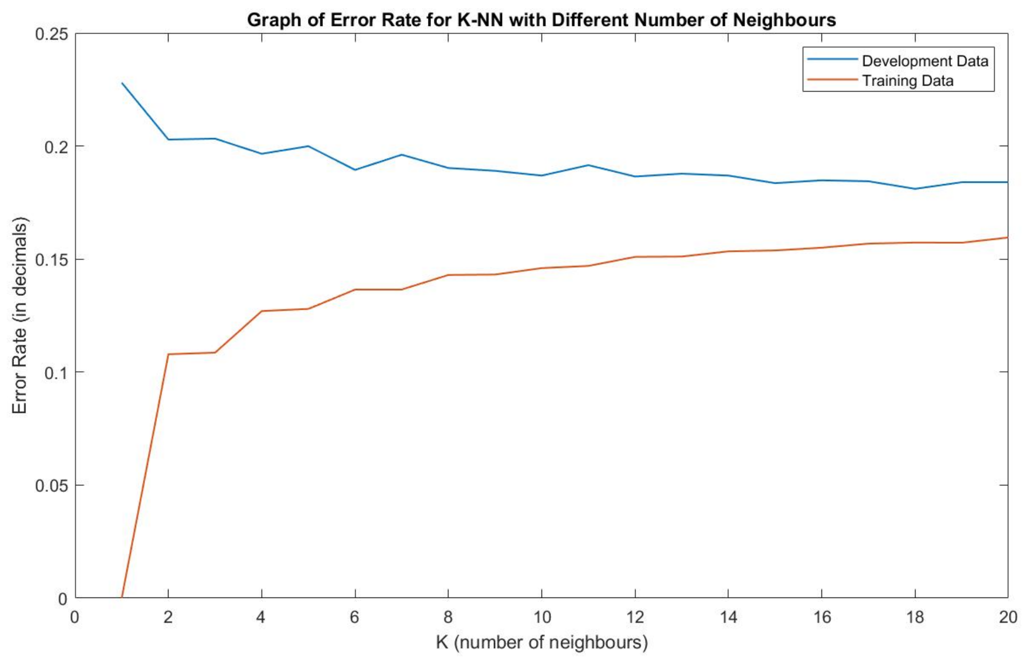
Question 2

(a) Training Error Rate with 1 nearest neighbour = 3.6002e-05
(b) Accuracy = 99.9964%

    K-nearest neighbour classifier with K=1 seems to have almost 100% of accuracy in classifying the samples. The model is built based on the training data, and the accuracy is tested on the same set of data. This classifier computes the distance between the query sample and all the other existing samples in the model and classify the query sample into the same group as the sample with shortest distance according to the inductive bias. This put the classifier into the risk of overfitting as it may be vulnerable to the noise in the data set. When there are some mistakes in getting the labels for the training data, the classifier will give a wrong classification on the different sets of data as that used for training the model.

Question 3

(c) Error rate computation for k=1-20



Question 4

Best K-NN model is the one with 18 neighbours.

Error of the model on the test data = 0.1615 = 16.15%

Accuracy of the model on test data = 83.85%

Conclusion

Based on the result of analysing two types of the classifiers, the model chosen to be presented to the project manager will be the decision tree model. It is chosen because of the higher accuracy achieved on the test data than the K-Nearest Neighbours (K-NN) model, which is 85.35% as compared to 83.85%. Besides, the classification process of decision tree takes shorter computation time than the K-NN. K-NN is the classifier that process every existing example in the model for analysing every new sample. It computes the distance between the new sample and the existing ones. This will be a great issue when we have a large data set. Hence, it is time consuming and it needs a larger storage space to store all the existing samples, whereas for decision tree, the previous data used for training the model and tuning the hyperparameter are not important to be retained once the best model had been built and published. All of these advantages posed by the decision tree makes it a better choice in this project.

In training a classifier, the presence of the missing value is crucial to be handled with care. The learning process throughout this coursework is done by removing the missing samples with "?" in all three sets of data. However, this is not reasonable and rational way of analysing the data and using it for modelling the classifiers, especially for the classification that based on the previous existing data, such as K-NN. The classification does not solely depend on the single variables. There are quite a number of variables used. The model trained will be not general enough to be released to the user by removing the sample. They will not be able to classify the new data received with "?". Modelling the classifiers by predicting the value of those samples with "?" will build a more useful and general model. This can be done by predicting the value of the sample to be around the value for those samples with the similar information for other variables. Interpolation method will be the example of the method for prediction.

Reference

Multiple plot in single layout

https://www.mathworks.com/help/matlab/creating_plots/combine-multiple-plots.html

One hot Encode

https://stackoverflow.com/questions/47666477/matlab-one-hot-encoding-convert-column-with-categoricals-into-several-columns

HWUM Statistical Machine Learning Lab material provided.

% Appendix

```
%%%%% function for deleting samples with missing value %%%%%

function Data=MissingValue(x,N)

nx=size(x);
nrow=nx(1,1);
ncolumn=nx(1,2);

Col2=table2array(x(:,2));
Col7=table2array(x(:,7));
Col14=table2array(x(:,14));

missing=Col2(N,1);
count=1;

for i=1:nrow

Miss=(Col2(i,1)==missing)|(Col7(i,1)==missing)| )|(Col14(i,1)==missing);
      if Miss==1
           count=count-1;
      else
          TrainData(count,:)=x(i,:);
      end
     count=count+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Balance Function %%%%%

function Dataset = Balance(Ratio)

Dataset='Unbalance';

if Ratio==50
    DataSet='Balance';
else
    DataSet=Dataset;
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Import and Save Data %%%%%

TrainData=MissingValue(TRData,8);
DevData=MissingValue(DData,14);
TestData=MissingValue(TEData,5);

save TrainData
save DevData
save TestData

TrainPredCategorical=[TrainData(:,2) TrainData(:,4:10) TrainData(:,14)];
save TrainPredCategorical
TrainPredNumeric=[TrainData(:,1) TrainData(:,3) TrainData(:,11:13)]
TrainPredNumeric =table2array(TrainPredNumeric);
save TrainPredNumeric
TrainLB=TrainData(:,15);
save TrainLB
```

```
DevPredCategorical=[DevData(:,2) DevData(:,4:10) DevData(:,14)];
save DevPredCategorical
DevPredNumeric=[ DevData(:,1) DevData(:,3) DevData(:,11:13)];
DevPredNumeric=table2array(DevPredNumeric);
save DevPredNumeric
DevLB=DevData(:,15);
save DevLB

TestPredCategorical=[TestData(:,2) TestData(:,4:10) TestData(:,14)];
save TestPredCategorical
TestPredNumeric= [TestData(:,1) TestData(:,3) TestData(:,11:13)];
TestPredNumeric=table2array(TestPredNumeric);
save TestPredNumeric
TestLB= TestData(:,15);
save TestLB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% PART (I) Statistics %%%%%

DataSize=size(TrainData);
nTrainEx=DataSize(1,1);
TrainLB=table2array(TrainLB);
nPositive=sum(TrainLB=='>50K');
nNegative=sum(TrainLB=='<=50K');

%%%%% Numeric Data %%%%%

Age=TrainPredNumeric(:,1);
FinalWeight=TrainPredNumeric(:,2);
CapGain=TrainPredNumeric(:,3);
CapLoss=TrainPredNumeric(:,4);
HourPerWeek=TrainPredNumeric(:,5);

mAge=mean(Age);
mFinalWeight=mean(FinalWeight);
mCapGain=mean(CapGain);
mCapLoss=mean(CapLoss);
mHourPerWeek=mean(HourPerWeek);

vAge=var(Age);
vFinalWeight=var(FinalWeight);
vCapGain=var(CapGain);
vCapLoss=var(CapLoss);
vHourPerWeek=var(HourPerWeek);

NFeature={'Age'; 'FinalWeight'; 'CapitalGain'; 'CapitalLoss'; 'Hour-per-
Week'};
M=[mAge; mFinalWeight; mCapGain; mCapLoss; mHourPerWeek];
V=[ vAge; vFinalWeight; vCapGain; vCapLoss; vHourPerWeek];

table(NFeature,M,V,'VariableNames',{'NumericFeature' 'Mean' 'Variance'});

%%%%% One-Hot Encoding for Training Data %%%%%

%Change the categorical predictors to 1-ok-K form
[Workclass, ~, indexWC] = unique(TrainPredCategorical(:,1));
WC = logical(accumarray([(1:nTrainEx).' indexWC], 1));

[Education, ~, indexED] = unique(TrainPredCategorical(:,2));
ED = logical(accumarray([(1: nTrainEx).' indexED], 1));

[MStatus ,~, indexMS] = unique(TrainPredCategorical(:,4));
```

```
MS= logical(accumarray([(1: nTrainEx).' indexMS], 1));

[Occupation, ~, indexOP] = unique(TrainPredCategorical(:,5));
OP = logical(accumarray([(1: nTrainEx).' indexOP], 1));

[Relation, ~, indexRS] = unique(TrainPredCategorical(:,6));
RS= logical(accumarray([(1: nTrainEx).' indexRS], 1));

[Race, ~, indexRC] = unique(TrainPredCategorical(:,7));
RC = logical(accumarray([(1: nTrainEx).' indexRC], 1));

[Sex, ~, indexSX] = unique(TrainPredCategorical(:,8));
SX = logical(accumarray([(1: nTrainEx).' indexSX],1));

[NCountry, ~, indexNC] = unique(TrainPredCategorical(:,9));
NC = logical(accumarray([(1: nTrainEx).' indexNC], 1));

WC=double(WC);  %to make it in matrix form
ED=double(ED);
MS=double(MS);
OP=double(OP);
RS=double(RS);
RC=double(RC);
SX=double(SX);
NC=double(NC);

CPredictors=horzcat(WC,ED,MS,OP,RS,RC,SX,NC);
Predictors=horzcat(CPredictors,TrainPredNumeric);
```

%%%%%% Categorical Data %%%%%

```
Workclass=table2array(Workclass);
Education=table2array(Education);
MStatus=table2array(MStatus);
Occupation=table2array(Occupation);
Relation=table2array(Relation);
Race=table2array(Race);
Sex=table2array(Sex);
NCountry=table2array(NCountry);

mWorkclass=mode(Workclass);
mEduction=mode(Education);
mMStatus=mode(MStatus);
mOccupation=mode(Occupation);
mRelation=mode(Relation);
mRace=mode(Race);
mSex=mode(Sex);
mNCountry=mode(NCountry);
mIncome=mode(TrainLB);
CFeature={'Workclass'; 'Education'; 'Marital Status'; 'Occuption';
'Relationship';'Race'; 'Sex'; 'NativeCountry'; 'Income' };
CMode=vertcat(mWorkclass, mEduction, mMStatus, mOccupation, mRelation,
mRace, mSex, mNCountry, mIncome);

table(CFeature,CMode,'VariableNames',{'CategoricalFeature' 'Mode'});
```

%%%%% Dataset Balance %%%%%

```
Ratio=nPositive/nNegative;

Dataset=Balance(Ratio)
```

%%%%% Bar Chart %%%%%

```
FWorkclass=countcats(categorical(indexWC));
FEducation=countcats(categorical (indexED));
FMStatus=countcats(categorical (indexMS));
FOccupation=countcats(categorical (indexOP));
FRelation=countcats(categorical (indexRS));
FRace=countcats(categorical (indexRC));
FSex=countcats(categorical (indexSX));
FNCountry=countcats(categorical (indexNC));

subplot(2,2,1);
bar(Workclass,FWorkclass);
subplot(2,2,2);
bar(Education,FEducation);
subplot(2,2,3);
bar(MStatus,FMStatus);
subplot(2,2,4);
bar(Occupation,FOccupation);

subplot(2,2,1);
bar(Relation,FRelation);
subplot(2,2,2);
bar(Race,FRace);
subplot(2,2,3);
bar(Sex,FSex);

bar(NCountry,FNCountry);

%%%%% Histogram %%%%%

subplot(3,2,1);
histogram(Age);
ylabel('Frequeny');
xlabel('Age');
title('Histogram of Age');

subplot(3,2,2);
histogram(FinalWeight);
ylabel('Frequeny');
xlabel('FinalWeight');
title('Histogram of FinalWeight');

subplot(3,2,3);
histogram(CapGain);
ylabel('Frequeny');
xlabel('Capital Gain');
title('Histogram of Capital Gain');

subplot(3,2,4);
histogram(CapLoss);
ylabel('Frequeny');
xlabel('Capital Loss');
title('Histogram of Capital Loss');

subplot(3,2,5);
histogram(HourPerWeek);
ylabel('Frequeny');
xlabel('Hours per Week');
title('Histogram of Hours per Week');

%%%%% Frequency Table %%%%%

Frequency=vertcat(FWorkclass,FEducation,FMStatus,FOccupation,FRelation,FRac
e,FSex,FNCountry);
```

```
Features=vertcat(unique(Workclass),unique(Education),unique(MStatus),unique
(Occupation),unique(Relation),unique(Race),unique(Sex),unique(NCountry));
table(Features,Frequency)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

%%%%% One-Hot Encoding for Training Data %%%%%

```
%Change the categorical predictors to 1-ok-K form
[Workclass, ~, indexWC] = unique(TrainPredCategorical(:,1));
WC = logical(accumarray([(1:nTrainEx).' indexWC], 1));

[Education, ~, indexED] = unique(TrainPredCategorical(:,2));
ED = logical(accumarray([(1: nTrainEx).' indexED], 1));

[MStatus ,~, indexMS] = unique(TrainPredCategorical(:,4));
MS= logical(accumarray([(1: nTrainEx).' indexMS], 1));

[Occupation, ~, indexOP] = unique(TrainPredCategorical(:,5));
OP = logical(accumarray([(1: nTrainEx).' indexOP], 1));

[Relation, ~, indexRS] = unique(TrainPredCategorical(:,6));
RS= logical(accumarray([(1: nTrainEx).' indexRS], 1));

[Race, ~, indexRC] = unique(TrainPredCategorical(:,7));
RC = logical(accumarray([(1: nTrainEx).' indexRC], 1));

[Sex, ~, indexSX] = unique(TrainPredCategorical(:,8));
SX = logical(accumarray([(1: nTrainEx).' indexSX],1));

[NCountry, ~, indexNC] = unique(TrainPredCategorical(:,9));
NC = logical(accumarray([(1: nTrainEx).' indexNC], 1));

WC=double(WC);  %to make it in matrix form
ED=double(ED);
MS=double(MS);
OP=double(OP);
RS=double(RS);
RC=double(RC);
SX=double(SX);
NC=double(NC);

CPredictors=horzcat(WC,ED,MS,OP,RS,RC,SX,NC);
Predictors=horzcat(CPredictors,TrainPredNumeric);
```

%%%%% One hot Encoding for Develop Data %%%%%

```
DevLB=table2array(DevLB);

DevSize=size(DevLB);
nDevEx=DevSize(1,1);

%Change the categorical predictors to 1-ok-K form
[DevWorkclass, ~, index] = unique(DevPredCategorical(:,1));
DWC = logical(accumarray([(1:nDevEx).' index], 1));

[DevEducation, ~, index] = unique(DevPredCategorical(:,2));
DED = logical(accumarray([(1: nDevEx).' index], 1));

[DevMStatus ,~, index] = unique(DevPredCategorical(:,4));
DMS= logical(accumarray([(1: nDevEx).' index], 1));
```

```
[DevOccupation, ~, index] = unique(DevPredCategorical(:,5));
DOP = logical(accumarray([(1: nDevEx).' index], 1));

[DevRelation, ~, index] = unique(DevPredCategorical(:,6));
DRS= logical(accumarray([(1: nDevEx).' index], 1));

[DevRace, ~, index] = unique(DevPredCategorical(:,7));
DRC = logical(accumarray([(1: nDevEx).' index], 1));

[DevSex, ~, index] = unique(DevPredCategorical(:,8));
DSX = logical(accumarray([(1: nDevEx).' index],1));

[DevNCountry, ~, index] = unique(DevPredCategorical(:,9));
DNC = logical(accumarray([(1: nDevEx).' index], 1));

Z=zeros(nDevEx,1);
%the predictors in the develop data does not consist of complete options as
training data
%create a column vector with entries of 0 to replace those options in some
predictors/features
%to make predictors for develop data consistent with the predictors used
for training model

DMS=double(DMS);
DMS=horzcat(DMS(:,1),Z,DMS(:,2:6));

DOP=double(DOP);
DOP=horzcat(DOP(:,1),Z,DOP(:,2:13));

DNC=double(DNC);
DNC=horzcat(DNC(:,1:14),Z,Z,DNC(:,15:18),Z,DNC(:,19:24),Z,DNC(:,25:31),Z,DN
C(:,32),Z,DNC(:,33:35));

DCPredictors=horzcat(DWC,DED,DMS,DOP,DRS,DRC,DSX,DNC);
DPredictors=horzcat(DCPredictors,DevPredNumeric);
```

%%%%% One hot Encoding for Test Data %%%%%

```
TestLB=table2array(TestData(:,15));

TestSize=size(TestLB);
nTestEx=TestSize(1,1);

[TestWorkclass, ~, index] = unique(TestPredCategorical(:,1));
TWC = logical(accumarray([(1:nTestEx).' index], 1));

[TestEducation, ~, index] = unique(TestPredCategorical(:,2));
TED = logical(accumarray([(1: nTestEx).' index], 1));

[TestMStatus ,~, index] = unique(TestPredCategorical(:,4));
TMS= logical(accumarray([(1: nTestEx).' index], 1));

[TestOccupation, ~, index] = unique(TestPredCategorical(:,5));
TOP = logical(accumarray([(1: nTestEx).' index], 1));

[TestRelation, ~, index] = unique(TestPredCategorical(:,6));
TRS= logical(accumarray([(1: nTestEx).' index], 1));

[TestRace, ~, index] = unique(TestPredCategorical(:,7));
TRC = logical(accumarray([(1: nTestEx).' index], 1));

[TestSex, ~, index] = unique(TestPredCategorical(:,8));
TSX = logical(accumarray([(1: nTestEx).' index],1));
```

```
[TestNCountry, ~, index] = unique(TestPredCategorical(:,9));
TNC = logical(accumarray([(1: nTestEx).' index], 1));

ZT=zeros(nTestEx,1);

TWC=double(TWC);
TED=double(TED);
TMS=double(TMS);
TOP=double(TOP);
TRS=double(TRS);
TRC=double(TRC);
TSX=double(TSX);
TNC=double(TNC);
TNC=horzcat(TNC(:,1:14),ZT,TNC(:,15:40));

TCPredictors=horzcat(TWC,TED,TMS,TOP,TRS,TRC,TSX,TNC);
TPredictors=horzcat(TCPredictors,TestPredNumeric);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Decision Tree %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Loss for split=1,3 %%%%%

nPredictor=(DataSize(1,2))-2 ; %number of predictor=total ncol-IncomeCol-
BinaryLabel

TrainedTree1=fitctree(Predictors ,TrainLB,'MaxNumSplits',1,'CrossVal',
'off');
%TrainLB is the information added in last col of the TrainData, binary
label
view(TrainedTree1,'Mode','graph');

LTrainedTree1= loss(TrainedTree1, Predictors, TrainLB);

TrainedTree2=fitctree(Predictors , TrainLB,
'MaxNumSplits',2,'CrossVal','off');
view(TrainedTree2,'Mode','graph');

LTrainedTree2=loss(TrainedTree2, Predictors, TrainLB);

TrainedTree3=fitctree(Predictors , TrainLB,
'MaxNumSplits',3,'CrossVal','off');
view(TrainedTree3,'Mode','graph');

LTrainedTree3=loss(TrainedTree3, Predictors, TrainLB);

Split=[1;2;3];
Loss=[LTrainedTree1; LTrainedTree2; LTrainedTree3];
table(Split,Loss)

%%%%% Error on Develop Data %%%%%

LTrainedTree4=loss(TrainedTree1,DPredictors, DevLB);
LTrainedTree5=loss(TrainedTree2,DPredictors, DevLB);
LTrainedTree6=loss(TrainedTree3,DPredictors, DevLB);

Split=[1;2;3];
Loss=[LTrainedTree4; LTrainedTree5; LTrainedTree6];
table(Split,Loss)

%%%%% Tuning Hyperparatmeter %%%%%
```

```
[DevErr,TrainErr,nSplit]=DError(DevLB,DPredictors,Predictors,TrainLB);

%%%%% Choose number of split and model %%%%%

CSplit=min((find(DevErr==min(DevErr)))*3-3); %formula used for the split
range
% find(DevErr==min(DevErr)) is to find the position of the data, not the
number of splits, as the split range is different
%CSplit=split chosen, with smallest error

CModel= fitctree(Predictors,TrainLB,'MaxNumSplits',CSplit, 'CrossVal',
'off');
view(CModel,'mode','graph')

%%%%% Error on Test Data %%%%%

TestError=TError(TestLB,TPredictors,CModel)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% K-NN Model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% standardize the numerical variables for all three sets of data
DevMean=mean(DevPredNumeric(:,1:5));
DMean=ones(length(DevLB),1)* DevMean;
SdDev=std(DevPredNumeric(:,1:5));
DevPredNumeric=((DevPredNumeric-DevMean)./DStd);
SDPredictors=[DPredictors(:,1:98) DevPredNumeric];

TrainMean=mean(TrainPredNumeric(:,1:5));
TMean=ones(length(TrainLB),1)* TrainMean;
SdTrain=std(TrainPredNumeric(:,1:5));
TrainPredNumeric=((TrainPredNumeric-TrainMean)./SdTrain);
SPredictors=[Predictors(:,1:98) TrainPredNumeric];


TestMean=mean(TestPredNumeric(:,1:5));
TEMean=ones(length(TestLB),1)* TestMean;
SdTest=std(TestPredNumeric(:,1:5));
TestPredNumeric=((TestPredNumeric-TestMean)./SdTest);
STPredictors=[TPredictors(:,1:98) TestPredNumeric];


%%%%% Loss for K=1 %%%%%

KnnModel1=fitcknn(SPredictors,TrainLB,'NumNeighbors',1);
KnnLoss=loss(KnnModel1,SPredictors,TrainLB);

%%%%% Tuning Hyperparameter %%%%%

KnnErr=KError(SDPredictors,DevLB,SPredictors,TrainLB);

%%%%% Choose the best k %%%%%

KNeighbor=find(KnnErr==min(KnnErr));
CKModel= fitcknn(SPredictors,TrainLB,'NumNeighbors',KNeighbor);

%%%%% Test Data %%%%%

KnnTestError=TError(TestLB,STPredictors,CKModel)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

%%%%%%%%%%%%%%%%%%% Function for Decision Tree and K-NN %%%%%%%%%%%%%%%%%%

%%%%% Decision Tree with Range of Number of Splits %%%%%

```
function
[DevErr,TrainErr,nSplit]=DError(DevLB,DPredictors,Predictors,TrainLB)

DevSize=size(DevLB);
nDevEx=DevSize(1,1);
TrainErr=zeros(50,1);

nSplit=zeros(50,1);
DevErr=zeros(50,1);

for j=1:50
    nSplit(j,1)=max(1,(3*j)-3);
    DevTree=fitctree(Predictors,TrainLB,'MaxNumSplits',nSplit(j,1),
'CrossVal', 'off');
    PreDevLB=predict(DevTree, DPredictors);

    for i=1:nDevEx
        if DevLB(i,1)~=PreDevLB(i,1)
        DevErr(j,1)=DevErr(j,1)+1;
        end
    end
    DevErr(j,1)=DevErr(j,1)/nDevEx;
    TrainErr(j,1)=loss(DevTree,Predictors,TrainLB);
end
table(nSplit,DevErr,TrainErr)
plot(nSplit,DevErr,"-")
hold on
plot(nSplit,TrainErr,"-")
hold off
```

%%%%% Error function for Range of Neighbours for K-NN %%%%%

```
function [K,KnnErr,KnnLoss]=KError(DPredictors,DevLB,Predictors,TrainLB)
DevSize=size(DevLB);
nDevEx=DevSize(1,1);

K=zeros(30,1);
KnnErr=zeros(30,1);
KnnLoss=zeros(30,1);

for j=1:30
    K(j,1)=j;
    KnnModel=fitcknn(Predictors, TrainLB,'NumNeighbors',
K(j,1),'standardize',1);
    PreKnnLB=predict(KnnModel, DPredictors);

    for i=1:nDevEx
        if DevLB(i,1)~=PreKnnLB(i,1)
        KnnErr(j,1)=KnnErr(j,1)+1;
        else
            KnnErr(j,1)=KnnErr(j,1);
        end
    end
    KnnErr(j,1)=KnnErr(j,1)/nDevEx;
    KnnLoss(j,1)=loss(DevTree,Predictors,TrainLB);
end
table(K,KnnErr,KnnLoss)
plot(K,KnnErr,"-")
```

```
hold on
plot(K,KnnLoss,"-")
hold off
```

%%%%% Test Error Function (Test Data) %%%%%

% This function for testing the accuracy on the test data is used for both model, as the chosen model is used as the input, simply change the model input when testing on different model

```
function  TestErr=TError(TestLB, TPredictors,CModel)

nTest=size(TestLB);
NTest=nTest(1,1);

TestErr=0;

TestModel=CModel

PreTestLB=predict(TestModel, TPredictors);

 for i=1:NTest
     if TestLB(i,1)~=PreTestLB(i,1)
        TestErr=TestErr+1;
     end
 end
TestErr=TestErr/NTest;
```