



P.E.S. COLLEGE OF ENGINEERING, MANDYA
(AN AUTONOMOUS INSTITUTE AFFILIATED TO VTU, BELAGAVI)



REPORT ON

**“Machine Learning-Based Prediction of Road
Accident Severity: A Comparative Analysis”**

Submitted by

RAKSHITH B S

4PS17ME076

Under the Guidance of

Mr. RANJITH.K. B.E,M.Tech

Assistant Professor.

Department of Mechanical Engineering

P.E.S. College of Engineering, Mandya.



**DEPARTMENT OF MECHANICAL ENGINEERING
P.E.S. COLLEGE OF ENGINEERING, MANDYA 2019-2**

ABSTRACT

Road accidents are a leading cause of mortality worldwide, with severe social and economic impacts. This study aims to analyze and predict the severity of road accidents using machine learning algorithms, including Random Forest, Decision Tree, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM). By evaluating various environmental, behavioral, and situational factors, we developed predictive models to estimate accident severity. The dataset, comprising 30 key features related to driver behavior, road conditions, and vehicle specifics, was preprocessed and analyzed to identify significant factors influencing accident outcomes. The Random Forest model achieved the highest accuracy of 80%, outperforming other algorithms in identifying high-risk conditions. This study highlights the potential of machine learning to improve road safety by providing accurate, data-driven insights, which can inform traffic management systems and enhance preventive measures.

INTRODUCTION

Road accidents are a major public health concern globally, leading to substantial loss of life, injuries, and economic consequences. According to the World Health Organization (WHO), traffic accidents are expected to become the fifth leading cause of death by 2030 if current trends continue. Given the complexity of factors influencing road accidents, such as environmental conditions, road infrastructure, driver behavior, and vehicle characteristics, accurately predicting accident severity has become a critical area of research in road safety and transportation planning.

Recent advancements in machine learning offer new avenues for predicting accident severity by analyzing large datasets encompassing diverse factors. Machine learning models can process and interpret complex relationships within data, providing insights into conditions that elevate accident risk. This study seeks to leverage machine learning algorithms to classify accident severity levels based on variables such as weather, road type, time of day, and driver demographics.

We apply and compare several widely used machine learning algorithms—including Random Forest, Decision Tree, KNN, and SVM—to identify the model that best predicts accident severity. By assessing the accuracy, precision, and recall of each model, this research contributes to a growing body of knowledge aimed at enhancing road safety through predictive analytics. The findings from this study could inform future traffic safety strategies, aiding in proactive accident prevention and improved measures

METHODOLOGY

Random Forest:

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random Forests are also very hard to beat in terms of performance.

Decision Tree:

A decision tree is one of the most frequently & widely used supervised machine learning algorithms that can perform both regression & classification tasks. The intuition behind the decision tree algorithm is simple, yet also powerful. For each attribute in the dataset, the decision tree algorithm forms a node, where the most important attribute is placed at the root node. For evaluation we start at the root node & work our way down the tree by following the corresponding node that meets our condition or decision. This process continues until a leaf node is reached, which contains the prediction or the outcome of the decision tree.

KNN:

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity has been used in statistical estimation and pattern recognition. A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor. KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry.

SVM:

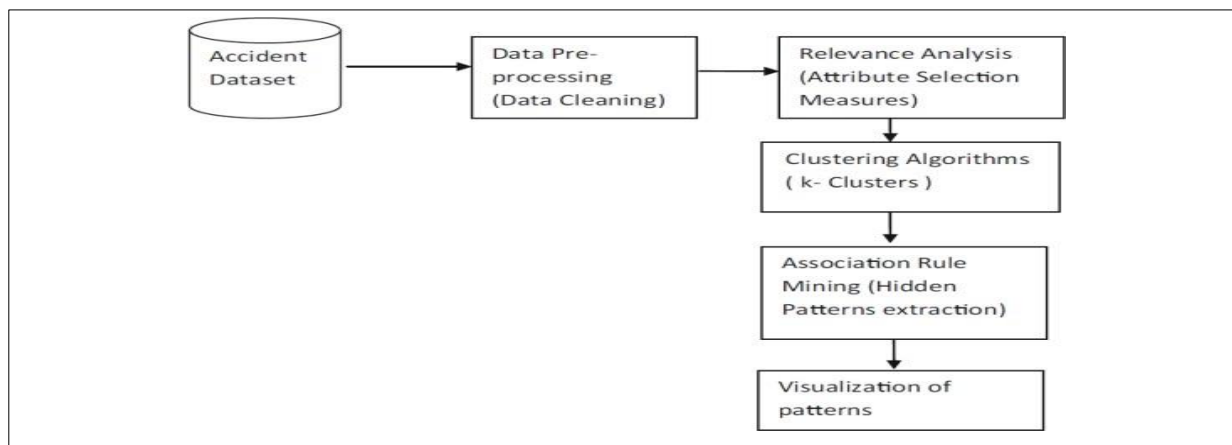
Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n- dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. It works really well with clear margin of separation. It is effective in high dimensional spaces. It is effective in cases where number of dimensions is greater than the number of samples.

PROPOSED ALGORITHM

Following steps are used in constructing system.

Database Creation:

In this step, A total of 30 attributes that focus on various criteria, such as accident- specific attributes, driver-specific attributes, circumstance-specific attributes, and other attributes given in the FIR report, form the input dataset.



Data Preprocessing:

After Database creation, Data pre-processing helps to remove noise, missing values, and inconsistencies. Missing values are replaced with NULL. Also, each attribute data is discretized in order to make it appropriate for further analysis.

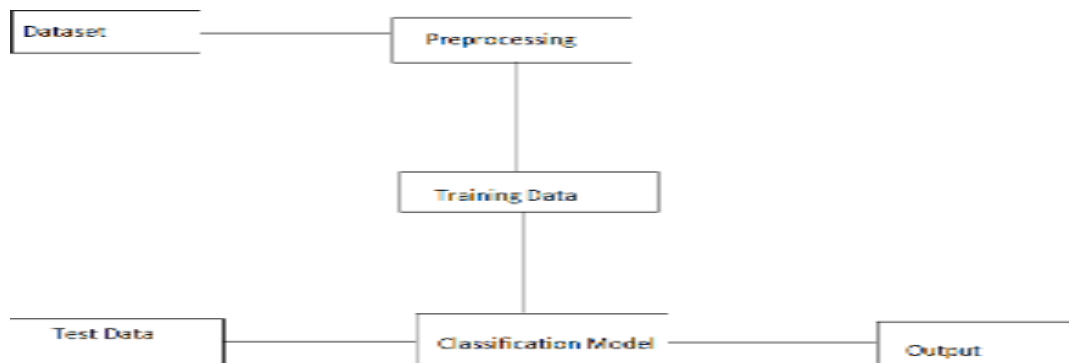
Data Cleaning:

In this step, the primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making. While it has been the focus of many researchers for several years, individual problems have been addressed separately. These include missing value imputation, outliers' detection, transformations, integrity constraints violations detection and repair, consistent query answering, deduplication, and many other related problems such as profiling and constraints mining.

Performance Evaluation:

1. Performance Measure - The performance measure is the way you want to evaluate a solution to the problem.
2. Test and Train Datasets- From the transformed data, you will need to select a test set and a training set.
3. Cross Validation.

PROPOSED ARCHITECTURE



The main objective of this research is to investigate the role of human, vehicle, and infrastructure-related factors in accident severity by applying machine learning techniques on road accident data. The steps include data cleaning, data transformation, relevance analysis, clustering, association rules generation, and finally performance evaluation

PERFORMANCE ANALYSIS

The performance is being analyzed by using database survey. Multiple reviews from users have been collected on the basis of simplified text.

Algorithms	Current Accuracy	Previous Accuracy
Random Forest	80 %	78.6%
Decision Tree	61%	83.6%
KNN	72%	77%
Naïve Bayes	73%	83.7%
Linear Regression	29%	76.8%

CONCLUSION

In this study, the technique of association rules with a large set of accident's data to identify the reasons of road accidents were used. The results show that this model could provide good predictions against traffic accident with 80% correct rate. It should be noted that due to the constraints of data and research condition, there are still some factors, such as engine capacity, traffic flows, accident location etc., not used in the model and they should be taken into account in future study. The results of this study can be used in vehicle safety assistance driving and provide early warnings and proposals for safe driving.

Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
train=pd.read_csv('Road_train.csv')
test=pd.read_csv('Road_test.csv')
train.head()

#train['src']='train'
#test['src']='test'
data=pd.concat([train,test],ignore_index=True)
data.shape

#cheking Missing values
data.isnull().sum()

plt.figure(1)

data['Time']=data.Time.str.replace(':',").astype('float64')
data['Date']=data.Date.str.replace('/',").astype('float64')
#data['src']=data.src.str.replace(' ','').astype('float64')

import numpy as np
import pandas as pd
cols=['Date','Time']
for col in cols:
    data[col]=data[col].astype(dtype=np.float64)
data.info()

new_data=data.dropna(axis=0,how='any')

new_data.isnull()
print(new_data.isnull().sum())#Total missing value
threshold=len(new_data)*.1
threshold
threshold=len(new_data)*.1
new_data.dropna(thresh=threshold,axis=1)#drops a colm that has < valus that of threshold
# axis=1 drop col
print(new_data.isnull().sum())

y=(new_data.Road_Type)
X=new_data.drop("Road_Type",axis=1)
#print(y)

new_data.head()

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```



```
from sklearn import datasets,linear_model
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn import linear_model
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA,TruncatedSVD
from sklearn import preprocessing
```

```
df_x=new_data.iloc[:,1:].values
df_y=new_data.iloc[:,0].values
```

```
x=new_data.iloc[:,1:].values
y=new_data.iloc[:,0].values
```

```
df = pd.DataFrame(data.drop(['accident_index'], axis = 1))
df.head()
```

```
df.dropna()
```

```
new_data=data.dropna(axis=0,how='any')
```

```
new_data.isnull()
print(new_data.isnull().sum())#Total missing value
threshold=len(new_data)*.1
threshold
threshold=len(new_data)*.1
new_data.dropna(thresh=threshold,axis=1)#drops a colm that has < valus that of threshold
# axis=1 drop col
print(new_data.isnull().sum())
```

```
y=(new_data.Road_Type)
X=new_data.drop("Road_Type",axis=1)
#print(y)
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

```
new_data.dtypes
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets,linear_model
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn import linear_model
```

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA,TruncatedSVD
from sklearn import preprocessing

df_x=new_data.iloc[:,1:].values
df_y=new_data.iloc[:,0].values

x=new_data.iloc[:,1:].values
y=new_data.iloc[:,0].values

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=4)

#rf=RandomForestClassifier(n_estimators=100)
#from sklearn import preprocessing
#le=preprocessing.LabelEncoder()
#rf.fit(x_train,y_train)
#pred=rf.predict(x_test)

reg=linear_model.LinearRegression()
reg.fit(x_train,y_train)
plt.style.use('fivethirtyeight')
plt.scatter(reg.predict(x_train),reg.predict(x_train)-y_train,color="green",s=10,label="Train
data')
plt.scatter(reg.predict(x_test),reg.predict(x_test)-y_test,color="blue",s=10,label="Train data')
plt.hlines(y=0,xmin=0,xmax=50,linewidth=2)
plt.legend(loc="upper right")
plt.title("Residual errors")
plt.show()

df = pd.DataFrame(data.drop(['accident_index'], axis = 1))
df.head()

df.dropna()

```

1.Random Forest

```

pca=PCA(n_components=16,whiten=True')
x=pca.fit(df_x).transform(df_x)
x_train,x_test,y_train,y_test=train_test_split(df_x,df_y,test_size=0.2,random_state=4)
rf=RandomForestClassifier(n_estimators=100)
rf.fit(x_train,y_train)
pred=rf.predict(x_test)
s=y_test
cnt=0
for i in range(len(pred)):

```

```
if(pred[i]==s[i]):  
    cnt=cnt+1  
    #principle component analysis  
cnt/float(len(pred))
```

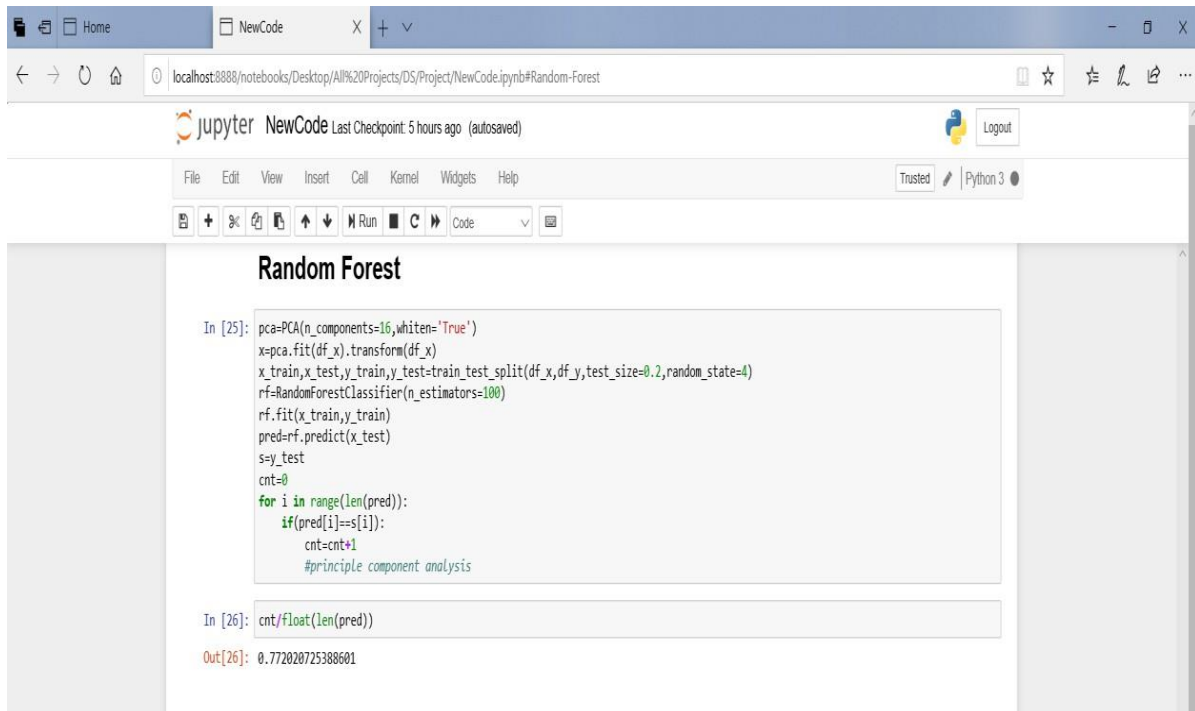
2.Decision Tree For Classification

```
from sklearn.tree import DecisionTreeClassifier  
classifier = DecisionTreeClassifier()  
classifier.fit(X_train, y_train)  
  
y_pred = classifier.predict(X_test)  
  
from sklearn.metrics import classification_report, confusion_matrix  
print(confusion_matrix(y_test, y_pred))  
print(classification_report(y_test, y_pred))
```

3.Decision Tree For Regression

```
from sklearn.tree import DecisionTreeRegressor  
regressor = DecisionTreeRegressor()  
regressor.fit(X_train, y_train)  
  
y_pred = regressor.predict(X_test)  
  
df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})  
df  
  
from sklearn import metrics  
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))  
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))  
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Output

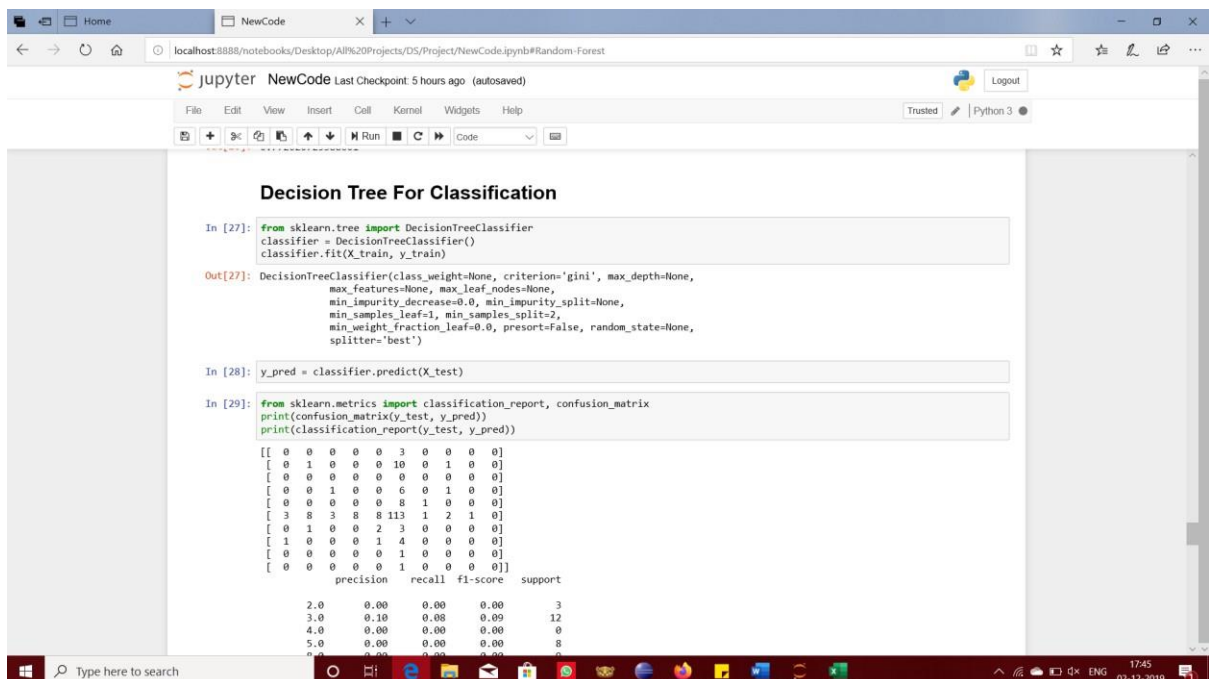


The screenshot shows a Jupyter Notebook titled "NewCode" with a Python 3 kernel. The notebook contains two code cells. The first cell, labeled "In [25]:", defines a PCA object with 16 components and whitening, fits it to the training data, splits the data into training and testing sets, trains a Random Forest classifier with 100 estimators, and makes predictions on the test set. A loop then counts the number of correct predictions. The second cell, labeled "In [26]:", calculates the accuracy as the count of correct predictions divided by the total number of predictions. The output of the second cell is "Out[26]: 0.772020725388601".

```
In [25]: pca=PCA(n_components=16,whiten='True')
x=pca.fit(df_x).transform(df_x)
x_train,x_test,y_train,y_test=train_test_split(df_x,df_y,test_size=0.2,random_state=4)
rf=RandomForestClassifier(n_estimators=100)
rf.fit(x_train,y_train)
pred=rf.predict(x_test)
s=y_test
cnt=0
for i in range(len(pred)):
    if(pred[i]==s[i]):
        cnt=cnt+1
    #principle component analysis

In [26]: cnt/float(len(pred))

Out[26]: 0.772020725388601
```



The screenshot shows a Jupyter Notebook titled "NewCode" with a Python 3 kernel. The notebook contains three code cells. The first cell, labeled "In [27]:", imports the DecisionTreeClassifier from sklearn.tree and fits it to the training data. The second cell, labeled "In [28]:", uses the trained classifier to make predictions on the test set. The third cell, labeled "In [29]:", imports the classification_report and confusion_matrix from sklearn.metrics and prints both. The output of the third cell shows a confusion matrix and a classification report.

```
In [27]: from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

Out[27]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False, random_state=None,
splitter='best')

In [28]: y_pred = classifier.predict(X_test)

In [29]: from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

[[ 0  0  0  0  0  3  0  0  0  0]
 [ 0  1  0  0  0 10  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  6  0  1  0  0]
 [ 0  0  0  0  0  0  1  0  0  0]
 [ 3  8  3  8  8 113  1  2  1  0]
 [ 0  1  0  0  2  3  0  0  0  0]
 [ 1  0  0  0  1  4  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0]]

precision    recall  f1-score   support

2.0      0.00      0.00      0.00         3
3.0      0.10      0.08      0.09        12
4.0      0.00      0.00      0.00         0
5.0      0.00      0.00      0.00         8
6.0      0.00      0.00      0.00         0
```

localhost:8888/notebooks/Desktop/All%20Projects/DS/Project/NewCode.ipynb#Random-Forest

Jupyter NewCode Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

4.0	0.00	0.00	0.00	0
5.0	0.00	0.00	0.00	8
8.0	0.00	0.00	0.00	9
9.0	0.76	0.77	0.76	147
11.0	0.00	0.00	0.00	6
19.0	0.00	0.00	0.00	6
21.0	0.00	0.00	0.00	1
90.0	0.00	0.00	0.00	1
avg / total	0.58	0.59	0.59	193

D:\python ide\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)

D:\python ide\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.
'recall', 'true', average, warn_for)

localhost:8888/notebooks/Desktop/All%20Projects/DS/Project/NewCode.ipynb#Random-Forest

Jupyter NewCode Last Checkpoint: 6 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Decision Tree For Regression

```
In [30]: from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
```

```
Out[30]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
presort=False, random_state=None, splitter='best')
```

```
In [31]: y_pred = regressor.predict(X_test)
```

```
In [32]: df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
df
```

```
Out[32]:
```

	Actual	Predicted
0	9.0	9.0
1	9.0	9.0
2	5.0	9.0
3	3.0	9.0
4	9.0	3.0
5	9.0	9.0
6	9.0	3.0
7	9.0	9.0
8	9.0	9.0
9	2.0	9.0
10	9.0	9.0
11	9.0	19.0

Type here to search

17:47 02-12-2019