Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2021-22

Experiment 10

(Mini Project)

Aim: Design a classifier to solve a specific problem in the given domain.

Tasks to be completed by the students:

Select a specific problem from any of the given domain areas, such as: Banking, Education, Insurance, Government, Media, Entertainment, Retail, Supply chain, Transportation, Logistics, Energy and Utility.

Task 1: Select appropriate dataset, describe the problem and justify the suitability of your dataset.

Task 2: Perform exploratory data analysis and pre-processing (if required).

Task 3: Apply appropriate machine learning algorithm to build a classify. Perform appropriate testing of your model.

Task 4: Submit a report in the given format.

- Introduction
- Data Description
- Data Analysis
- Reason to select machine learning model
- Algorithm
- Result Analysis
- Conclusion and Future Scope.
- Python notebook

Task5: Presentation

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)
Department of Computer Science and Engineering (Data Science)

Report on Mini Project

Machine Learning -I (DJ19DSC402)

AY: 2021-22

Students' Career Prediction

ADITYA POTDAR: 60009200027

SARVAGYA SINGH: 60009200030

Guided By

Dr. Kriti Srivastava

CHAPTER 1: INTRODUCTION

Competition in today's society is heavily multiplying day by day. It is too heavy in the present day's technical world. So as to compete and reach the goal students need to be planned and organized from the initial stages of their education. So it is very important to constantly evaluate their performance, identify their interests and evaluate how close they are to their goal, and assess whether they are on the right path that directs toward their target. This helps them in improving themselves, motivating themselves to a better career path if their capabilities are not up to the mark to reach their goal, and pre-evaluate themselves before going to the career peak point.

CHAPTER 2: DATA DESCRIPTION

The first 9 fields of the dataset contain the academic score of the candidate in OS, Algorithms, Programming Concepts, Software Engineering, Computer Networks, Electronics, Computer Architecture, Mathematics, and Communication skills.

The next column is the hours worked per day by the candidate. Then there are three columns in which the candidates are rated in Logical Quotient, Coding, and Public speaking respectively. The number of hackathons also plays an important role in the students' computer science careers. If students have self-learning capability and can work a long time before the system, then the chances of excelling in their career will increase. Extra courses, certifications, and workshops are like adding a cherry on the cake for the candidate. Giving talent tests and olympiads would do anyone a world of good. There are many types of roles like Database administrator, Business Process Analyst, Developer, Testing Manager, Networks Manager, Data scientist, and so on. All these roles require some prerequisite knowledge in them to be placed in them. Other fields like Hard/smart worker, introvert, and behavior also play an important role in their career.

CHAPTER 3: DATA ANALYSIS

Collecting the data is one task and making that data useful is another vital task. Data collected from various means will be in an unorganized format and there may be a lot of null values, invalid data values, and unwanted data.

- Cleaning all these data and replacing them with appropriate or approximate data and removing null and missing data and replacing them with some fixed alternate values are the basic steps in pre-processing of data.
- Even data collected may contain completely garbage values. It may not be in the exact format or way that is meant to be.
- All such cases are verified and replaced with alternate values to make data meaningful and useful for further processing.
- Data is kept in an organized format.

CHAPTER 4: REASON TO SELECT MACHINE LEARNING MODEL

- Decision tree is used for career choice prediction.
  Calculated OOB Out of Bag Score: This metric is the accuracy of examples xi using all the trees in the random forest ensemble for which it was omitted during training. Use of forests of trees to evaluate the importance of features on an artificial classification task.
- SVM: A kind of generalised linear classifier that classifies data based on supervised learning.

CHAPTER 5: ALGORITHM

- **SVM :-**

  SVM denotes Support Vector Machine. It is a supervised machine learning algorithm that is generally used for both regression and classification types of problems. The main applications of this can be found in various classification problems. The typical procedure of the algorithm is first each data item is plotted in an n-dimensional space, where n is the number of features and the value of each feature is the value of that particular coordinate. The next step is to classify by getting the hyper-plane that separates the two classes very finely.

- **DECISION TREE (RANDOM FOREST):-**

  Decision Trees are extremely popular and one of the simple and easy to implement machine learning classification problems. Decision trees laid the basic foundation for many advanced algorithms like bagging, gradient boosting, and random forest. The commonly used decision trees are CART, C4.5, C5, and ID3. A node denotes an input variable (X) and a split on that variable, assuming the variable is numerical. The leaf which is also called the terminal node of the tree possesses an output variable (y) which is vital for the prediction.

  The typical scenario that a decision tree follows is first selecting a root node. Calculate information gain or entropy for each of the nodes before the split. Select the node that has more information gain or less entropy. Further, split the node and reiterate the process. The process is iterated until there is no possibility to split or the entropy is minimum. Entropy is the metric to measure the uncertainty or randomness of data. Information gain is the metric that measures how much entropy is reduced before to after a split.

CHAPTER 6: RESULT ANALYSIS

The data is trained and tested with all three algorithms and out of all SVM gave more accuracy. As SVM gave the highest accuracy, all further data predictions are to be followed with SVM.

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

A more powerful web application can be developed where inputs are not given directly instead student parameters are taken by evaluating students through various evaluations and examining. Technical, analytical, logical, memory-based, psychometry and general awareness, interests, and skill-based tests can be designed and parameters are collected through them so that results will be certainly accurate and the system will be more reliable to use.

CHAPTER 8: PYTHON NOTEBOOK

Students' Career Prediction

By

- **Aditya Potdar**
- **Sarvagya Singh**

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

```
df = pd.read_csv('career_pred.csv')
df
```

	Academic percentage in Operating Systems	Percentage in Algorithms	Percentage in Programming Concepts	Percentage in Software Engineering	Percentage in Computer Networks	Percentage in Electronics Subjects	Percentage in Computer Architecture	Percer Mathema
0	69	63	78	87	94	94	87	
1	78	62	73	60	71	70	73	
2	71	86	91	87	61	81	72	
3	76	87	60	84	89	73	62	
4	92	62	90	67	71	89	73	
...	...	...	...	...	...	...	...	
19995	83	67	62	63	81	74	90	
19996	80	69	83	87	82	66	66	
19997	83	70	80	87	64	85	69	
19998	68	87	91	88	66	74	61	
19999	73	77	74	84	70	65	92	

20000 rows × 39 columns

```
# Data
data = df.iloc[:,:-1].values
label = df.iloc[:,-1]
```

```
#Label Encoding: COnverting To Numeric values
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder = LabelEncoder()

for i in range(14,38):
    data[:,i] = labelencoder.fit_transform(data[:,i])
```

```
#Normalizing the data
from sklearn.preprocessing import Normalizer
data1 = data[:,:14]
normalized_data = Normalizer().fit_transform(data1)

data2 = data[:,14:]
df1 = np.append(normalized_data,data2,axis=1)
```

```
#Combining into a dataset
df2=df.iloc[:,:-1]
dataset = pd.DataFrame(df1,columns=df2.columns)
dataset
```

	Academic percentage in Operating Systems	Percentage in Algorithms	Percentage in Programming Concepts	Percentage in Software Engineering	Percentage in Computer Networks	Percentage in Electronics Subjects	Percentage in Computer Architecture	Percer Mathema
0	0.28509	0.260299	0.322276	0.359461	0.388383	0.388383	0.359461	0.34
1	0.34998	0.278189	0.327545	0.269215	0.318571	0.314085	0.327545	0.37
2	0.295012	0.357339	0.378115	0.361494	0.253461	0.336563	0.299168	0.29
3	0.328025	0.375503	0.258967	0.362554	0.384135	0.315077	0.2676	0.37
4	0.397157	0.267649	0.388523	0.289234	0.306502	0.384206	0.315136	0.30
...	...	...	...	...	...	...	...	
19995	0.366575	0.29591	0.273827	0.278244	0.357742	0.326826	0.397491	0.37
19996	0.34342	0.2962	0.356299	0.37347	0.352006	0.283322	0.283322	0.39
19997	0.343084	0.289348	0.330683	0.359618	0.264546	0.351351	0.285214	0.38
19998	0.294792	0.377161	0.394502	0.381496	0.286122	0.320804	0.264446	0.37
19999	0.311409	0.328472	0.315675	0.358334	0.298611	0.277282	0.392461	0.31

20000 rows × 38 columns

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
# For label
label = df.iloc[:,-1]
original=label.unique()
label=label.values
label2 = labelencoder.fit_transform(label)
y=pd.DataFrame(label2,columns=["Suggested Job Role"])
numeric=y["Suggested Job Role"].unique()
Y = pd.DataFrame({'Suggested Job Role':original, 'Associated Number':numeric})
```

```
data
```

```
array([[69, 63, 78, ..., 0, 1, 0],
       [78, 62, 73, ..., 0, 0, 1],
       [71, 86, 91, ..., 0, 0, 1],
       ...,
       [83, 70, 80, ..., 0, 0, 1],
       [68, 87, 91, ..., 1, 1, 0],
       [73, 77, 74, ..., 0, 1, 0]], dtype=object)
```

```
Y
```

	Suggested Job Role	Associated Number
0	Database Developer	7
1	Portal Administrator	18
2	Systems Security Administrator	28
3	Business Systems Analyst	2
4	Software Systems Engineer	25
5	Business Intelligence Analyst	1
6	CRM Technical Developer	4
7	Mobile Applications Developer	14
8	UX Designer	32
9	Quality Assurance Associate	21
10	Web Developer	33
11	Information Security Analyst	11
12	CRM Business Analyst	3
13	Technical Support	31
14	Project Manager	20
15	Information Technology Manager	13
16	Programmer Analyst	19
17	Design & UX	9
18	Solutions Architect	26
19	Systems Analyst	27
20	Network Security Administrator	16
21	Data Architect	5

Decision Tree

Random Forest Technique is used to find the accuracy and important features to be used while implementing Decision Tree.

The OOB_score is computed as the number of correctly predicted rows from the out-of-bag sample

```
from sklearn.ensemble import RandomForestClassifier
rf_model=RandomForestClassifier(n_estimators=1000,max_features=2,oob_score=True)
X=dataset.copy()
rf_model.fit(X,df['Suggested Job Role'])
print("OOB Accuracy")
print(rf_model.oob_score_)
```

```
OOB Accuracy
0.04185
```

```
feature = X.columns
imp=np.empty(len(feature))
i=0
for feat,imp_val in zip(feature,rf_model.feature_importances_):
    #print(feature,imp)
    imp[i]=imp_val
    i=i+1

indices = np.argsort(imp)

for i in indices:
    print(feature[i],imp[i])
```

```
Taken inputs from seniors or elders 0.012811110960687402
Management or Technical 0.012924976494005603
worked in teams ever? 0.01292798791026999
hard/smart worker 0.012954982848684684
Salary Range Expected 0.012973628388032394
Salary/work 0.013037797208379685
In a Realtionship? 0.01304455484994843
interested in games 0.013142303634780219
self-learning capability? 0.013151088886122377
```

```
Introvert 0.013189163469575018
Job/Higher Studies? 0.01331518127305238
Extra-courses did 0.013342581200612902
olympiads 0.0133587217205776
Gentle or Tuff behaviour? 0.013402734354447925
talenttests taken? 0.013427815699384444
can work long time before system? 0.013477369841440363
reading and writing skills 0.019543007743343114
memory capability score 0.01991554572370892
interested career area  0.02795169642690302
workshops 0.03032881847013935
certifications 0.03103903260024671
Type of company want to settle in? 0.03167248057039675
Interested subjects 0.03186024306470999
Interested Type of Books 0.036855049887493024
hackathons 0.03824832513087548
Logical Quotient rating 0.040071043193723656
Percentage in Algorithms 0.040107747429633266
Percentage in Programming Concepts 0.04010932531176585
Percentage in Electronics Subjects 0.040114438250857976
Percentage in Mathematics 0.040122435431165816
Percentage in Software Engineering 0.04013370595645964
Hours working per day 0.04015894876631692
coding skills rating 0.040158978187190136
Percentage in Computer Architecture 0.04018244401449325
Percentage in Computer Networks 0.04019440952156639
Percentage in Communication skills 0.040195487158761006
public speaking points 0.04023603281678687
Academic percentage in Operating Systems 0.04031880560346133
```

Since all values are almost close, we can not eliminate any one of the features.

```
#Decision Tree Classifier

from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
```

```
DT_model = tree.DecisionTreeClassifier()
DT_model = DT_model.fit(X_train, y_train)
DT_model
```

```
DecisionTreeClassifier()
```

```
#Generate Decision Tree Graph
features=['Acedamic percentage in Operating Systems', 'percentage in Algorithms',
       'Percentage in Programming Concepts',
       'Percentage in Software Engineering', 'Percentage in Computer Networks',
       'Percentage in Electronics Subjects',
       'Percentage in Computer Architecture', 'Percentage in Mathematics',
       'Percentage in Communication skills', 'Hours working per day',
       'Logical quotient rating', 'hackathons', 'coding skills rating',
       'public speaking points', 'can work long time before system?',
       'self-learning capability?', 'Extra-courses did', 'certifications',
       'workshops', 'talenttests taken?', 'olympiads',
       'reading and writing skills', 'memory capability score',
       'Interested subjects', 'interested career area ', 'Job/Higher Studies?',
       'Type of company want to settle in?',
       'Taken inputs from seniors or elders', 'interested in games',
       'Interested Type of Books', 'Salary Range Expected',
       'In a Realtionship?', 'Gentle or Tuff behaviour?',
       'Management or Technical', 'Salary/work', 'hard/smart worker',
       'worked in teams ever?', 'Introvert']
with open('Dtree3.dot','w') as f:
    f=tree.export_graphviz(DT_model,feature_names=features,out_file=f)
```

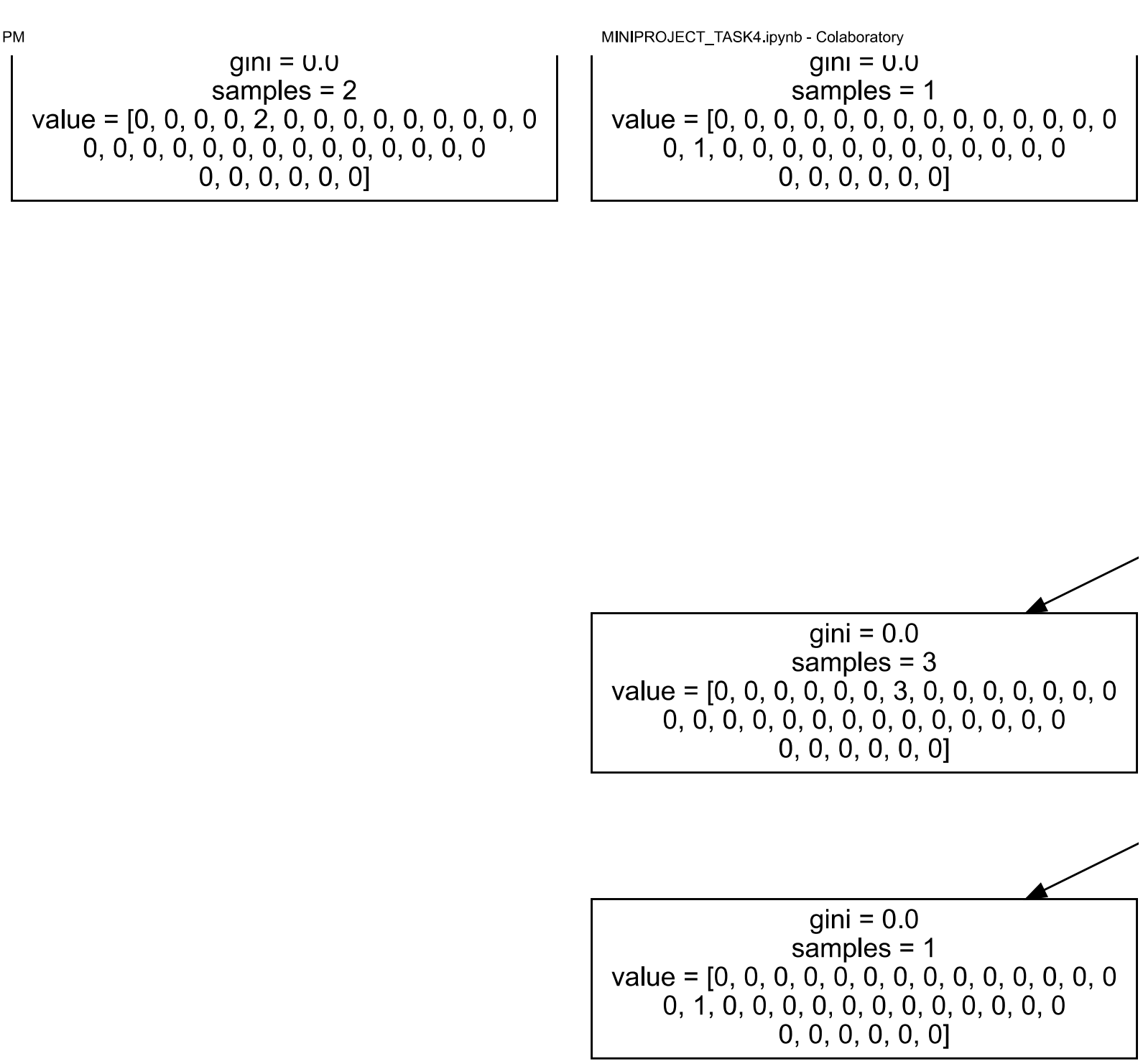
```
r=tree.export_text(DT_model,feature_names=features)
print(r)
```

```
     |--- Percentage in Electronics Subjects <= 0.44
     |   |--- percentage in Algorithms <= 0.44
     |   |   |--- percentage in Algorithms <= 0.44
     |   |   |   |--- Acedamic percentage in Operating Systems <= 0.44
     |   |   |   |   |--- coding skills rating <= 0.04
     |   |   |   |   |   |--- percentage in Algorithms <= 0.39
```

```
|   |   |   |   |   |   |--- public speaking points <= 0.04
|   |   |   |   |   |   |   |--- Percentage in Electronics Subjects <= 0.31
|   |   |   |   |   |   |   |   |--- Logical quotient rating <= 0.00
|   |   |   |   |   |   |   |   |   |--- Acedamic percentage in Operating Systems <= 0.37
|   |   |   |   |   |   |   |   |   |   |--- Percentage in Computer Architecture <= 0.38
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 13
|   |   |   |   |   |   |   |   |   |   |--- Percentage in Computer Architecture >  0.38
|   |   |   |   |   |   |   |   |   |   |   |--- class: 14
|   |   |   |   |   |   |   |   |   |--- Acedamic percentage in Operating Systems >  0.37
|   |   |   |   |   |   |   |   |   |   |--- Hours working per day <= 0.03
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |--- Hours working per day >  0.03
|   |   |   |   |   |   |   |   |   |   |   |--- class: 22
|   |   |   |   |   |   |   |   |--- Logical quotient rating >  0.00
|   |   |   |   |   |   |   |   |   |--- Acedamic percentage in Operating Systems <= 0.24
|   |   |   |   |   |   |   |   |   |   |--- class: 28
|   |   |   |   |   |   |   |   |   |--- Acedamic percentage in Operating Systems >  0.24
|   |   |   |   |   |   |   |   |   |   |--- Hours working per day <= 0.03
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 35
|   |   |   |   |   |   |   |   |   |   |--- Hours working per day >  0.03
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 33
|   |   |   |   |   |   |   |   |--- Percentage in Electronics Subjects >  0.31
|   |   |   |   |   |   |   |   |   |--- Percentage in Software Engineering <= 0.41
|   |   |   |   |   |   |   |   |   |   |--- percentage in Algorithms <= 0.39
|   |   |   |   |   |   |   |   |   |   |   |--- Percentage in Software Engineering <= 0.41
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 50
|   |   |   |   |   |   |   |   |   |   |   |--- Percentage in Software Engineering >  0.41
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |   |--- percentage in Algorithms >  0.39
|   |   |   |   |   |   |   |   |   |   |   |--- interested career area  <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |   |   |--- interested career area  >  0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 11
|   |   |   |   |   |   |   |   |   |--- Percentage in Software Engineering >  0.41
|   |   |   |   |   |   |   |   |   |   |--- percentage in Algorithms <= 0.34
|   |   |   |   |   |   |   |   |   |   |   |--- Interested Type of Books <= 9.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 9
|   |   |   |   |   |   |   |   |   |   |   |--- Interested Type of Books >  9.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 9
|   |   |   |   |   |   |   |   |   |   |--- percentage in Algorithms >  0.34
|   |   |   |   |   |   |   |   |   |   |   |--- workshops <= 2.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |   |--- workshops >  2.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 8
|   |   |   |   |   |   |   |--- public speaking points >  0.04
|   |   |   |   |   |   |   |   |--- Percentage in Software Engineering <= 0.35
|   |   |   |   |   |   |   |   |   |--- Percentage in Programming Concepts <= 0.25
|   |   |   |   |   |   |   |   |   |   |--- Percentage in Software Engineering <= 0.32
|   |   |   |   |   |   |   |   |   |   |   |--- Interested subjects <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 21
|   |   |   |   |   |   |   |   |   |   |   |--- Interested subjects >  1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 9
```

```
from sklearn import tree
import graphviz
dot_data = tree.export_graphviz(DT_model, out_file=None)
graph = graphviz.Source(dot_data)
graph
```

X[18] <= 6.0
gini = 0.444
samples = 3
value = [0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0]


gini = 0.0
samples = 2
value = [0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 3
value = [0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0]

```
# Prediction
y_pred = DT_model.predict(X_test)
y_test_arr = y_test['Suggested Job Role']
Final = pd.DataFrame({'Predicted' : y_pred, 'Actual' : y_test_arr})
Final.reset_index()
```

	index	Predicted	Actual
0	19778	24	24
1	4376	29	24
2	10188	15	10
3	9887	18	5
4	4441	5	9
...	...	...	...
3995	13123	14	20
3996	18706	16	17
3997	7274	23	25
3998	16155	26	0
3999	16712	28	26

4000 rows × 3 columns

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
print("accuracy=",accuracy*100)
```

```
accuracy= 2.875
```

```
#Number of correct predicted
comparison_column = np.where(Final['Predicted'] == Final['Actual'], 1, 0)
Number_of_correct_predictions = list(comparison_column).count(1)
print("Number_of_correct_predictions : ",((Number_of_correct_predictions)/4000)*100,"%")
accuracy_DT = Number_of_correct_predictions
```

```
Number_of_correct_predictions :  2.875 %
```

```
y_train=y_train['Suggested Job Role']
y_train
```

```
18660    16
10831     4
5511     28
3543     11
12340     3
         ..
9372     31
7291     19
17728     5
7293      6
17673    31
Name: Suggested Job Role, Length: 16000, dtype: int64
```

Support Vector Machine

```
# Run svm with default hyper parameters
# Default hyperparameter means C=1.0, kernel=rbf and gamma=auto among other parameters
# import SVC classifier
```

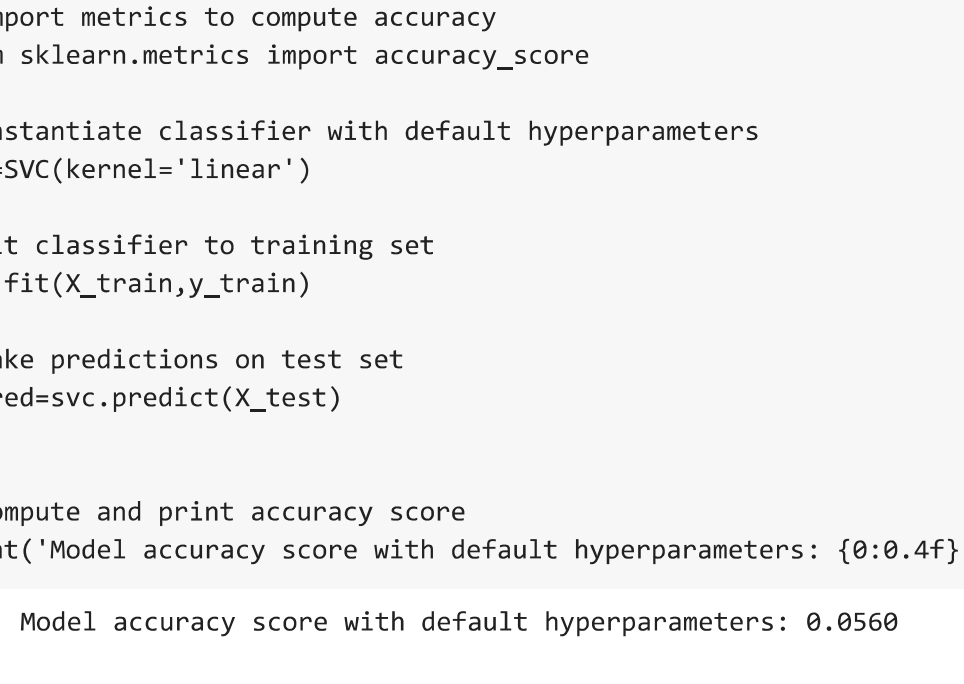
```
from sklearn.svm import SVC

# import metrics to compute accuracy
from sklearn.metrics import accuracy_score

# instantiate classifier with default hyperparameters
svc=SVC()

# fit classifier to training set
svc.fit(X_train,y_train)

# make predictions on test set
y_pred=svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with default hyperparameters: {0:0.4f}'. format(accuracy_score(y_test, y_pred)*100))
```

```
Model accuracy score with default hyperparameters: 5.6000
```

```
# Run svm with default hyper parameters
# Default hyperparameter means C=1.0, kernel=linear and gamma=auto among other parameters
# import SVC classifier
from sklearn.svm import SVC

# import metrics to compute accuracy
from sklearn.metrics import accuracy_score

# instantiate classifier with default hyperparameters
svc=SVC(kernel='linear')

# fit classifier to training set
svc.fit(X_train,y_train)

# make predictions on test set
y_pred=svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with default hyperparameters: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with default hyperparameters: 0.0560
```

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
Name = ['Decision Tree', 'SVM']
accuracies = [2.825,5.6]
ax.bar(Name,accuracies)
ax.set_ylabel('Accuracies')
ax.set_xlabel('Techniques')
plt.show()
```



Since, SVM has highest accuracy, it is best suited for this given data.

Prediction On Own Dataset Using SVM

```
x_new = ['3','5','4','3','5','4','3','5','2','2','5','2','5','4','2','5','5','3','2','5','5','4','2','3','4','3','2','1','5','6','3',
new_pred  = svc.predict([x_new])
print("Prediction : {}".format(Y[Y['Associated Number']==new_pred[0]]['Suggested Job Role']))
```

```
Prediction : 20    Network Security Administrator
Name: Suggested Job Role, dtype: object
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but SVC was fitte
  "X does not have valid feature names, but"
```