Placement prediction_Code

anonymous marking enabled

Submission date: 16-Jul-2023 06:09PM (UTC-0700)
Submission ID: 2132164746
File name: Placement_prediction_Code.txt (8.93K)
Word count: 609
Character count: 7587

model.py (python file)

```
import joblib
import pandas as CSE3D
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

Team74 = CSE3D.read_csv('student_data.csv')

Team74.head()

Team74.shape

Team74.info()

Team74.isnull().sum()

Team74.describe()

Team74.columns
```

```
Team74 = Team74.drop(['sl_no'],axis=1)


print('gender:',Team74['gender'].unique())
print('Stream:',Team74['Stream'].unique())
print('Self Learning Capability:',Team74['Self Learning Capability'].unique())
print('Extra_Course:',Team74['Extra_Courses'].unique())
print('Internship:',Team74['Internship'].unique())
print('status:',Team74['status'].unique())


Team74['gender'] = Team74['gender'].map({'M':0,'F':1})
Team74['Stream'] = Team74['Stream'].map({'CSE':2,'ECE':1,'Other':0})
Team74['Self Learning Capability'] = Team74['Self Learning
Capability'].map({'Yes':1,'No':0})
Team74['Extra_Courses'] = Team74['Extra_Courses'].map({'Yes':1,'No':0})
Team74['Internship'] = Team74['Internship'].map({'Yes':1,'No':0})
Team74['status'] = Team74['status'].map({'Placed':1,'Not Placed':0})


Team74.head()


Team = Team74.iloc[:, 0:10]
Team


plt.scatter(Team['TechnicalSkill_perc'], Team['status'])
```

```
plt.show()


plt.scatter(Team['Self Learning Capability'], Team['status'])
plt.show()


plt.scatter(Team['Coding_perc'], Team['status'])
plt.show()


plt.scatter(Team['Extra_Courses'], Team['status'])
plt.show()


plt.scatter(Team['Communication_perc'], Team['status'])
plt.show()


plt.scatter(Team['Internship'], Team['status'])
plt.show()


plt.scatter(Team['LogicalReasoning_perc'], Team['status'])
plt.show()


a = Team.drop('status',axis=1)
b= Team['status']
```

```
a.columns
```

```
import seaborn as sns

#Checking for Outliers
fig, axs = plt.subplots(ncols=3,nrows=3,figsize=(20,10))

index = 0

axs = axs.flatten()

for k,v in a.items():

    sns.boxplot(b=v, ax=axs[index])

    index+=1


plt.tight_layout(pad=0.3, w_pad=0.5,h_pad = 4.5) # for styling by giving padding


# checking distributions of all features
fig, axs = plt.subplots(ncols=3,nrows=3,figsize=(20,10))

index = 0

axs = axs.flatten()

for k,v in a.items():

    sns.distplot(v, ax=axs[index])

    index+=1



plt.tight_layout(pad=0.3, w_pad=0.2,h_pad = 4.5)
```

```
from sklearn.model_selection import train_test_split

a_train,a_test,b_train,b_test=train_test_split(a,b,test_size=0.20,random_state=42)

a_train

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

logReg = LogisticRegression()
logReg.fit(a_train,b_train)

SVM = svm.SVC()
SVM.fit(a_train,b_train)

KNN=KNeighborsClassifier()
KNN.fit(a_train,b_train)
```

```
DT=DecisionTreeClassifier()

DT.fit(a_train,b_train)


RandFor=RandomForestClassifier()

RandFor.fit(a_train,b_train)


GradBoost=GradientBoostingClassifier()

GradBoost.fit(a_train,b_train)


b_pred1 = logReg.predict(a_test)

b_pred2 = SVM.predict(a_test)

b_pred3 = KNN.predict(a_test)

b_pred4 = DT.predict(a_test)

b_pred5 = RandFor.predict(a_test)

b_pred6 = GradBoost.predict(a_test)


from sklearn.metrics import accuracy_score


ACC1=accuracy_score(b_test,b_pred1)

ACC2=accuracy_score(b_test,b_pred2)

ACC3=accuracy_score(b_test,b_pred3)

ACC4=accuracy_score(b_test,b_pred4)

ACC5=accuracy_score(b_test,b_pred5)
```

```
ACC6=accuracy_score(b_test,b_pred6)


ACCURACY = CSE3D.DataFrame({'Models':['LogisticRegression',
                                      'SuppotVectorMachine',
                                      'KNeighborsClassifier',
                                      'DecisionTreeClassifier',
                                      'RandomForestClassifier',
                                      'GradientBoostingClassifier'],
          'ACC':[ACC1*100,
                 ACC2*100,
                 ACC3*100,
                 ACC4*100,
                 ACC5*100,
                 ACC6*100]})


ACCURACY


model1=GradientBoostingClassifier()
model1.fit(a,b)


new_data = CSE3D.DataFrame({
    'gender':0,
    'Stream':2,
```

```
    'TechnicalSkill_perc':70,
    'Self Learning Capability':1,
    'Coding_perc':50,
    'Extra_Courses':1,
    'Communication_perc':50,
    'Internship':1,
    'LogicalReasoning_perc':80
},index=[0])

pp=model1.predict(new_data)
prob=model1.predict_proba(new_data)
if pp==1:
    print('Placed')
    print(f"You will be placed with probability of {prob[0][1]:.2f}")
else:
    print("Not-placed")
    print(f"You will not placed with probability of {prob[0][0]:.2f}")
```

6

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
# Split the dataset into input features (job descriptions) and target variable (job roles)
Group= Team74.iloc[:,10:15]
Group


print('certifications:',Group['certifications'].unique())
print('workshops',Group['workshops'].unique())
print('Interested subjects',Group['Interested subjects'].unique())


p = Group.drop('Suggested Job Role',axis=1)
q = Group['Suggested Job Role']


p.columns


Group['Suggested Job Role'].unique()


# Split the data into training and testing sets
p_train, p_test, q_train, q_test = train_test_split(p, q, test_size=0.2, random_state=42)


p_train['combined_features'] = p_train.apply(lambda row: '
'.join(row.values.astype(str)), axis=1)
p_test['combined_features'] = p_test.apply(lambda row: ' '.join(row.values.astype(str)),
axis=1)
```

```
vectorizer = TfidfVectorizer()
p_train_vectors = vectorizer.fit_transform(p_train['combined_features'])
p_test_vectors = vectorizer.transform(p_test['combined_features'])

print(p_train_vectors.shape)
print(q_train.shape)

# Train a logistic regression classifier
model2 = GradientBoostingClassifier()
model2.fit(p_train_vectors, q_train)

# Make predictions on the test set
q_pred = model2.predict(p_test_vectors)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(q_test, q_pred)
print("Accuracy:", accuracy)

new_interests = ["app development   web technologies    hacking "]
new_interests_vector = vectorizer.transform(new_interests)
predicted_job_role = model2.predict(new_interests_vector)
print("Predicted Job Role:", predicted_job_role)
```

```
joblib.dump(model1, 'model_pp1.pkl')

joblib.dump(model2, 'model_pp2.pkl')

joblib.dump(vectorizer, "vectorizer.joblib")
```

app.py

```
import streamlit as st
from sklearn.feature_extraction.text import TfidfVectorizer
import joblib
import numpy as np

model1 = joblib.load('model_pp1.pkl')

model2 = joblib.load('model_pp2.pkl')

loaded_vectorizer = joblib.load("vectorizer.joblib")

st.title("Welcome to Placement Prediction")

Name = st.text_input("Name:")

a=st.radio(

    "Gender(Male[1]/Female[0]:",

    (1,0))

b=st.radio(
```

```
    "Stream(Other[0]/ECE[1]/CSE[2]):",
    (0,1,2))
c = int(st.number_input("Technical Skill Percentage:", min_value=0, max_value=100,
step=1))
d=st.radio(
    "Self-Learning Capability(yes[1]/No[0]):",
    (1,0))
e = int(st.number_input("Coding_perc:", min_value=0, max_value=100, step=1))
f=st.radio(
    "Extra Courses(yes[1]/No[0]):",
    (1,0))
g = int(st.number_input("Communication_perc:", min_value=0, max_value=100,
step=1))
h=st.radio(
    "Internship(yes[1]/No[0]):",
    (1,0))
i = int(st.number_input("Logical Reasoning perc:", min_value=0, max_value=100,
step=1))

o1=['shell programming','machine learning','app development','python',
 'r programming','information security','hadoop','distro making',
 'full stack']
j1 = st.selectbox(
```

```
        'Certifications',o1)
o2 = ['cloud computing','database security','web technologies','data science',
 'testing','hacking','game development','system designing']
j2 = st.selectbox(
        'Workshops',o2)
o3 = ['cloud computing','networks','hacking','Computer Architecture',
 'programming','parallel computing','IOT','data engineering',
 'Software Engineering','Management']

j3 = st.selectbox(
        'Interested Subjects',o3)




btn=st.button("predict")
new_data=np.array([a,b,c,d,e,f,g,h,i]).reshape(1,-1)



if btn:
    pred=model1.predict(new_data)
    prob=model1.predict_proba(new_data)
```

```
if pred==1:
    st.write('Placed')
    st.write(f"You will be placed with probability of {prob[0][1]:.2f}")
    new_job_description = f"{j1.title()} {j2.title()} {j3.title()}"
    new_job_vector = loaded_vectorizer.transform([new_job_description])
    predicted_job_role = model2.predict(new_job_vector)
    st.write("Suggested Job Role:", predicted_job_role)
else:
    st.write('Not-Placed')
    st.write("You need to improve your Skills !")
    st.write("Here are some websites for your Reference...")
    st.write("https://www.geeksforgeeks.org/")
    st.write("https://www.javatpoint.com/")
```

Placement prediction_Code

ORIGINALITY REPORT

30%	25%	10%	25%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Taylor's Education Group Student Paper	6%
2	Submitted to NIIT University Student Paper	3%
3	Submitted to University of Hertfordshire Student Paper	3%
4	robotics.hochschule-rhein-waal.de Internet Source	3%
5	Submitted to University of East London Student Paper	2%
6	www.coursehero.com Internet Source	2%
7	Submitted to Monash University Student Paper	2%
8	dev.to Internet Source	1%
9	Submitted to Indian School of Business Student Paper	1%

10	Submitted to University of Missouri, Kansas City Student Paper	1 %
11	tkhan11.github.io Internet Source	1 %
12	tlsdmstn56.github.io Internet Source	1 %
13	thecleverprogrammer.com Internet Source	1 %
14	medium.com Internet Source	1 %
15	tanthiamhuat.files.wordpress.com Internet Source	1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		