ML-MAJOR-NOV-ML11B3

Problem Statement : For a given dataset (problem) which is the best classification algorithm (as per accuracy),Pick up a dataset of your choice or the one attached with mail (Preferable) .Ask any two questions on the dataset of your choice and provide answers for the same.¶

Task 1:Reading and Inspection:

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```python
df = pd.read_csv(r'C:\Users\shash\Desktop\Placement_Data_Full_Class.csv')
df
```

Out[2]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

215 rows × 15 columns

In [3]:
```python
#Listing columns in dataframe
list(df.columns.values)
```

Out[3]:
```
['sl_no',
 'gender',
 'ssc_p',
 'ssc_b',
 'hsc_p',
 'hsc_b',
 'hsc_s',
 'degree_p',
 'degree_t',
 'workex',
 'etest_p',
 'specialisation',
 'mba_p',
 'status',
 'salary']
```

In [4]:
```python
#printing number of rows and columns in data frame
nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
There are 215 rows and 15 columns
```

In [5]:
```python
#Let's take a quick look at what the data looks like:
df.head(10)
```

Out[5]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.00	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.50	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.00	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.00	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.80	Mkt&Fin	55.50	Placed	425000.0
5	6	M	55.00	Others	49.80	Others	Science	67.25	Sci&Tech	Yes	55.00	Mkt&Fin	51.58	Not Placed	NaN
6	7	F	46.00	Others	49.20	Others	Commerce	79.00	Comm&Mgmt	No	74.28	Mkt&Fin	53.29	Not Placed	NaN
7	8	M	82.00	Central	64.00	Central	Science	66.00	Sci&Tech	Yes	67.00	Mkt&Fin	62.14	Placed	252000.0
8	9	M	73.00	Central	79.00	Central	Commerce	72.00	Comm&Mgmt	No	91.34	Mkt&Fin	61.29	Placed	231000.0
9	10	M	58.00	Central	70.00	Central	Commerce	61.00	Comm&Mgmt	No	54.00	Mkt&Fin	52.21	Not Placed	NaN

In [6]:
```python
df.shape
```

Out[6]:
```
(215, 15)
```

Task 2:Cleaning the Data

Subtask 2.1:Inspect Null Values

In [7]:
```python
# #Code for column-wise null count
df.isnull().sum(axis=0).sort_values(ascending= False)
```

Out[7]:
```
salary            67
status             0
mba_p              0
specialisation     0
etest_p            0
workex             0
degree_t           0
degree_p           0
hsc_s              0
hsc_b              0
hsc_p              0
ssc_b              0
ssc_p              0
gender             0
sl_no              0
dtype: int64
```

In [8]:
```python
#Code for row-wise null count
df.isnull().sum(axis=1).sort_values(ascending= False)
```

Out[8]:
```
214    1
29     1
155    1
34     1
158    1
      ..
125    0
124    0
123    0
122    0
0      0
Length: 215, dtype: int64
```

In [9]:
```python
#Code for column-wise null percentages
df.isnull().sum(axis=0).sort_values(ascending= False)/len(df)
```

Out[9]:
```
salary            0.311628
status            0.000000
mba_p             0.000000
specialisation    0.000000
etest_p           0.000000
workex            0.000000
degree_t          0.000000
degree_p          0.000000
hsc_s             0.000000
hsc_b             0.000000
hsc_p             0.000000
ssc_b             0.000000
ssc_p             0.000000
gender            0.000000
sl_no             0.000000
dtype: float64
```

In [10]:
```python
#to find null values
df.isnull().sum()
```

Out[10]:
```
sl_no              0
gender             0
ssc_p              0
ssc_b              0
hsc_p              0
hsc_b              0
hsc_s              0
degree_p           0
degree_t           0
workex             0
etest_p            0
specialisation     0
mba_p              0
status             0
salary            67
dtype: int64
```

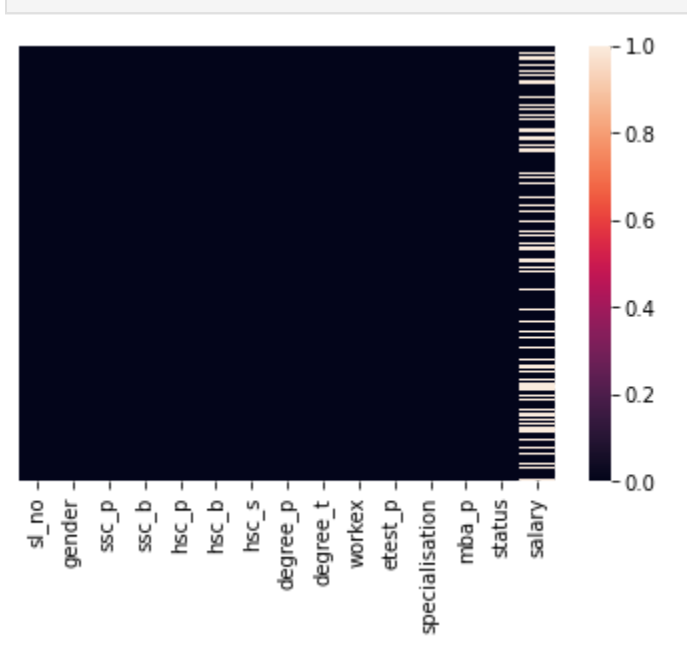
Subtask 2.2:Handling The Problem

In [11]:
```python
df.isnull().any()
```

Out[11]:
```
sl_no             False
gender            False
ssc_p             False
ssc_b             False
hsc_p             False
hsc_b             False
hsc_s             False
degree_p          False
degree_t          False
workex            False
etest_p           False
specialisation    False
mba_p             False
status            False
salary             True
dtype: bool
```

In [12]:
```python
#if dataset is totally black,it means dataset is perfect
import seaborn as sns
sns.heatmap(df.isnull(), yticklabels=False)

plt.show()
```



In [13]:
```python
df.columns
```

Out[13]:
```
Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
       'status', 'salary'],
      dtype='object')
```

In [14]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sl_no           215 non-null    int64
 1   gender          215 non-null    object
 2   ssc_p           215 non-null    float64
 3   ssc_b           215 non-null    object
 4   hsc_p           215 non-null    float64
 5   hsc_b           215 non-null    object
 6   hsc_s           215 non-null    object
 7   degree_p        215 non-null    float64
 8   degree_t        215 non-null    object
 9   workex          215 non-null    object
 10  etest_p         215 non-null    float64
 11  specialisation  215 non-null    object
 12  mba_p           215 non-null    float64
 13  status          215 non-null    object
 14  salary          148 non-null    float64
dtypes: float64(6), int64(1), object(8)
memory usage: 25.3+ KB
```

In [15]:
```python
df['salary'].fillna(value=0, inplace=True)
```

In [16]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sl_no           215 non-null    int64
 1   gender          215 non-null    object
 2   ssc_p           215 non-null    float64
 3   ssc_b           215 non-null    object
 4   hsc_p           215 non-null    float64
 5   hsc_b           215 non-null    object
 6   hsc_s           215 non-null    object
 7   degree_p        215 non-null    float64
 8   degree_t        215 non-null    object
 9   workex          215 non-null    object
 10  etest_p         215 non-null    float64
 11  specialisation  215 non-null    object
 12  mba_p           215 non-null    float64
 13  status          215 non-null    object
 14  salary          215 non-null    float64
dtypes: float64(6), int64(1), object(8)
memory usage: 25.3+ KB
```

Subtask 2.3:Identifying Duplicates

In [17]:
```python
#to identify the duplicate data
df.duplicated()
```

Out[17]:
```
0      False
1      False
2      False
3      False
4      False
       ...
210    False
211    False
212    False
213    False
214    False
Length: 215, dtype: bool
```

In [18]:
```python
df.duplicated().sum()
```

Out[18]:
```
0
```

since there is no duplicate elements in dataset . the dataset is clean and can be used on any EDA questions.

Task 3 : EDA Questions

Ques1. To get placed with highest salary which degree should be Opted?

In [19]:
```python
df.degree_t.value_counts()
```

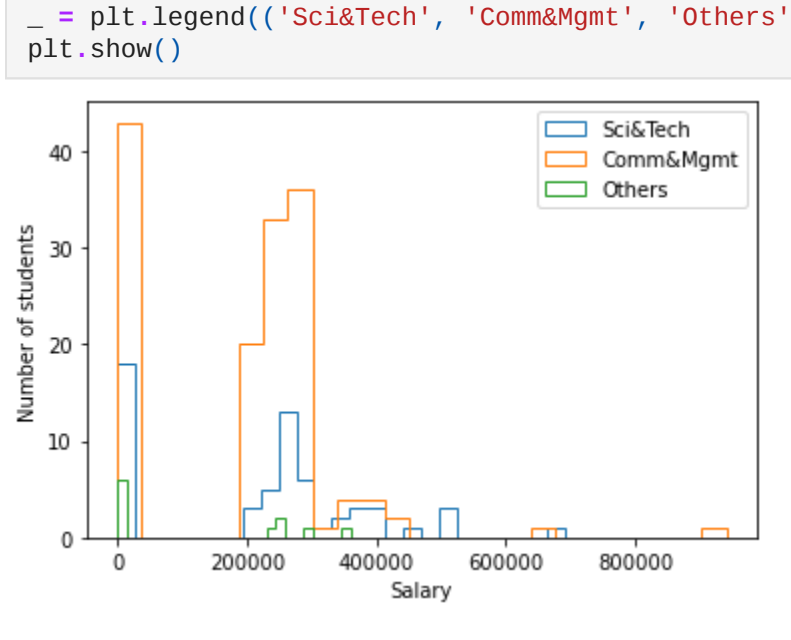
Out[19]:
```
Comm&Mgmt    145
Sci&Tech      59
Others        11
Name: degree_t, dtype: int64
```

In [20]:
```python
dfST=np.array(df[(df.degree_t=='Sci&Tech')].salary)
dfCM=np.array(df[(df.degree_t=='Comm&Mgmt')].salary)
dfO=np.array(df[(df.degree_t=='Others')].salary)


_=plt.hist(dfST,bins=25,histtype='step')
_=plt.hist(dfCM,bins=25,histtype='step')
_=plt.hist(dfO,bins=25,histtype='step')

plt.xlabel('Salary')
plt.ylabel('Number of students')

_ = plt.legend(('Sci&Tech', 'Comm&Mgmt', 'Others'))
plt.show()
```



As we can see the degree "Comm&Mgmt" has the hightest salary.So, "Comm&Mgmt" should be opted for highest salary!!

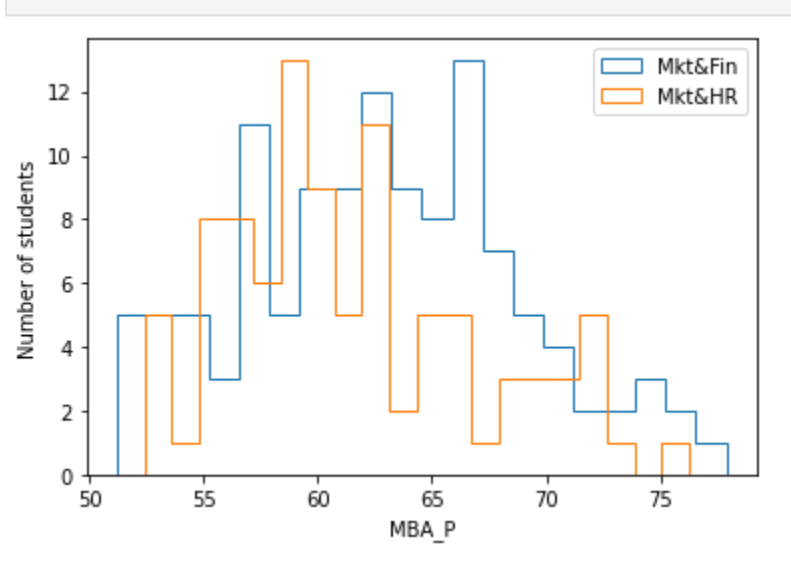
Ques2. Which specialisation has the highest "mba_p" ?

In [21]:
```python
df.specialisation.value_counts()
```

Out[21]:
```
Mkt&Fin    120
Mkt&HR      95
Name: specialisation, dtype: int64
```

In [22]:
```python
dfMF=np.array(df[(df.specialisation=='Mkt&Fin')].mba_p)
dfMH=np.array(df[(df.specialisation=='Mkt&HR')].mba_p)
```

In [23]:
```python
_=plt.hist(dfMF,bins=20,histtype='step')
_=plt.hist(dfMH,bins=20,histtype='step')
_=plt.legend(('Mkt&Fin', 'Mkt&HR'))
plt.xlabel('MBA_P')
plt.ylabel('Number of students')
plt.show()
```



As we can see the "mba_p" of "Mkt&Fin" specialsation is the most.So,student with "Mkt&Fin" specialsation has the highest "mba_p"!!

Label Encoding the columns

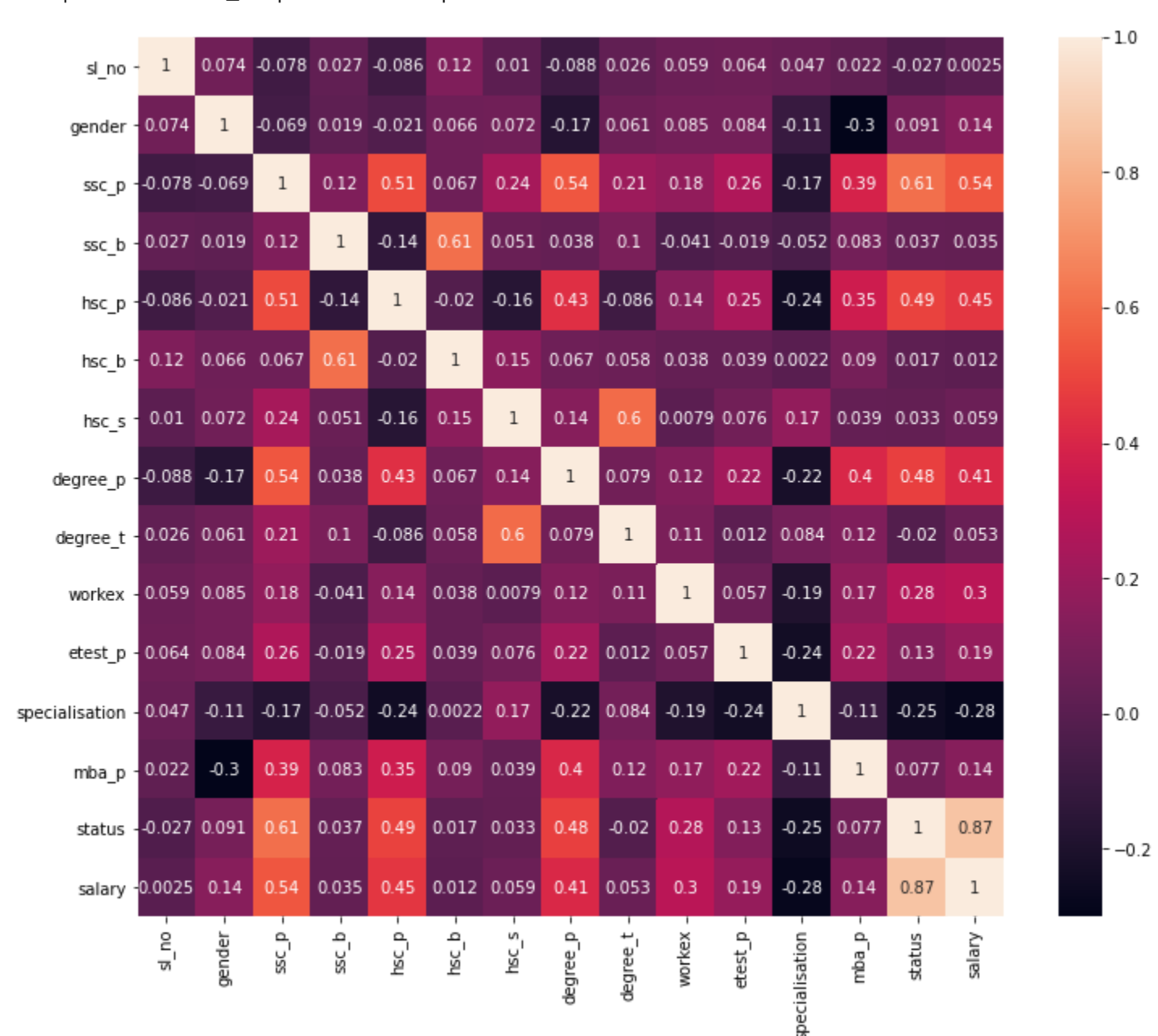
In [24]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['status']=le.fit_transform(df['status'])
df['hsc_b']=le.fit_transform(df['hsc_b'])
df['workex']=le.fit_transform(df['workex'])
df['gender']=le.fit_transform(df['gender'])
df['ssc_b']=le.fit_transform(df['ssc_b'])
df['hsc_s']=le.fit_transform(df['hsc_s'])
df['degree_t']=le.fit_transform(df['degree_t'])
df['specialisation']=le.fit_transform(df['specialisation'])
```

Checking correlation (Feature Selection)

In [25]:
```python
import seaborn as sb
plt.rcParams['figure.figsize'] = 12,10


sb.heatmap(df.corr(),annot=True)
```

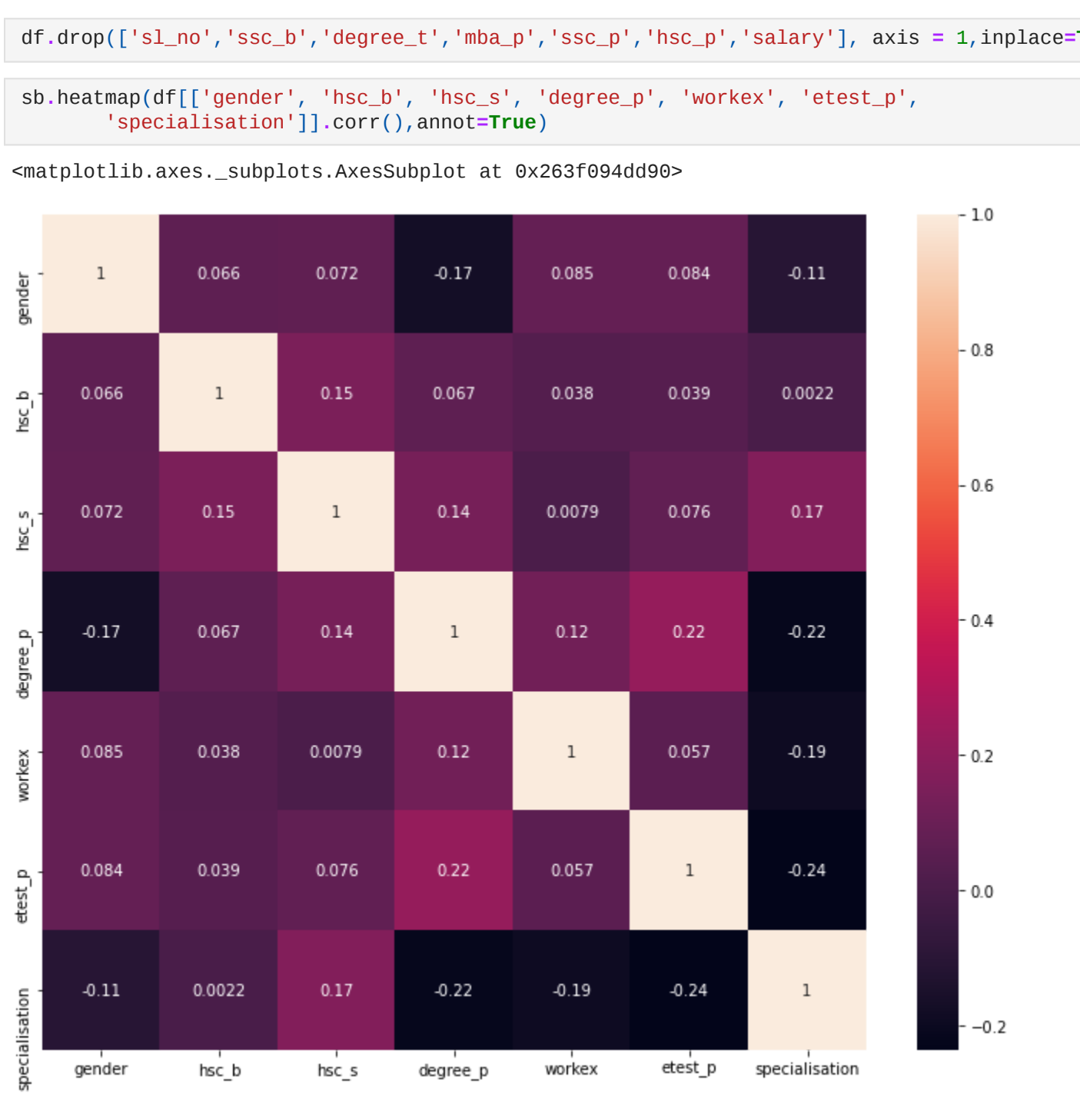
Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x263f0078e50>
```



In [26]:
```python
df.drop(['sl_no','ssc_b','degree_t','mba_p','ssc_p','hsc_p','salary'], axis = 1,inplace=True)
```

In [27]:
```python
sb.heatmap(df[['gender', 'hsc_b', 'hsc_s', 'degree_p', 'workex', 'etest_p',
       'specialisation']].corr(),annot=True)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x263f094dd90>
```



In [28]:
```python
df.columns
```

Out[28]:
```
Index(['gender', 'hsc_b', 'hsc_s', 'degree_p', 'workex', 'etest_p',
       'specialisation', 'status'],
      dtype='object')
```

Assigning Independent and Dependent Variables

In [29]:
```python
X = df[['gender', 'hsc_b', 'hsc_s', 'degree_p', 'workex', 'etest_p',
       'specialisation']]
y = df[['status']]
```

Train & Test Split

In [30]:
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=0)
```

Ensemble Machine learning Modelling (3 Classification Algorithms)

Algorithm 1: SVM

In [31]:
```python
from sklearn.svm import SVC
svc=SVC()
```

In [32]:
```python
svc.fit(X_train,y_train)
```

```
C:\Users\shash\anaconda3.1\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the sha
pe of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Out[32]:
```
SVC()
```

In [33]:
```python
y_pred1 = svc.predict(X_test)
```

Accuracy calculation of SVM Algorithm:

In [34]:
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_pred1, y_test)
```

Out[34]:
```
0.6851851851851852
```

Algorithm 2: Logistic Regression

In [35]:
```python
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [36]:
```python
logreg.fit(X_train, y_train)
y_pred2 = logreg.predict(X_test)
```

```
C:\Users\shash\anaconda3.1\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the sha
pe of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Accuracy calculation of Logistic Regression Algorithm:

In [37]:
```python
logreg.score(X_test, y_test)
```

Out[37]:
```
0.7777777777777778
```

Algorithm 3: KNN

In [38]:
```python
from sklearn.neighbors import KNeighborsClassifier
KN=KNeighborsClassifier()
```

In [39]:
```python
KN.fit(X_train,y_train)
```

```
<ipython-input-39-e258bd0a7847>:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example usin
g ravel().
  KN.fit(X_train,y_train)
```

Out[39]:
```
KNeighborsClassifier()
```

In [40]:
```python
y_pred3=KN.predict(X_test)
```

Accuracy calculation of KNN Algorithm:

In [41]:
```python
accuracy_score(y_test,y_pred3)
```

Out[41]:
```
0.7592592592592593
```

Summary

We see the best model is Logistic Regression with 77.77% accuracy, followed by KNN Classification with 75.92% accuracy and Support Vector Machine (SVM) with 68.51% accuracy.

In [ ]: