IBM_Capstone_Project

June 16, 2020

1 Campus Recruitment

2 Problem Statement

Campus placement is becoming highly competitive and there is immense load on colleges. This puts great pressures on the students if they are studying in some reputed college as the fear of not getting placed is constantly haunting them due to shallow fall in economy of country due to COVID-19. Here I will try to address the key dependencies of credentials earned from class 10th to current degree that would affect he chances of placement. Some key points undertaken are: -

1. Choice of board in class 10th and 12th to get placed.
2. Does gender effects the placements stats?
3. Work Experience, and internships effects.
4. What factors are responsible for not getting placed?
5. How does stream effects placement?

At the end a model will be trained to perform predictive analysis

The dataset can be accesed from dataset

3 Importing important libraries and reading data

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

[2]:
```
df = pd.read_csv("/home/baap/capstone_projects/IBM/dataset/placement/
 ↪placement_data.csv")
df.head()
```

[2]:
```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0      1      M  67.00   Others  91.00   Others  Commerce     58.00
1      2      M  79.33  Central  78.33   Others   Science     77.48
2      3      M  65.00  Central  68.00  Central      Arts     64.00
3      4      M  56.00  Central  52.00  Central   Science     52.00
4      5      M  85.80  Central  73.60  Central  Commerce     73.30

   degree_t workex  etest_p specialisation  mba_p      status    salary
0  Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0
1  Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2 Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
3  Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN
4 Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0
```

[3]:
```
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
```

[4]:
```
df.isnull().sum()
```

[4]:
```
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
```

4 Data Exploration

From info of the data, we can see the salary has only 148 entries and thus 67 entries are null value. This is due to the fact htat a guy is placed or not, thus for data cleansing ,we will remove the null values with some values helpful for us.(here zero)

[5]:
```
#Replacing all the null values with zero
df['salary'].fillna(0, inplace = True)
```

[6]:
```
df.head()
```

[6]:
```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0      1      M  67.00   Others  91.00   Others  Commerce     58.00
1      2      M  79.33  Central  78.33   Others   Science     77.48
2      3      M  65.00  Central  68.00  Central      Arts     64.00
3      4      M  56.00  Central  52.00  Central   Science     52.00
4      5      M  85.80  Central  73.60  Central  Commerce     73.30

    degree_t workex  etest_p specialisation  mba_p      status    salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
```

```
3   Sci&Tech      No      66.0          Mkt&HR   59.43  Not Placed       0.0
4  Comm&Mgmt      No      96.8         Mkt&Fin   55.50      Placed  425000.0
```

Now we will be replacing the string values with integer values for our understanding, such as status will be rplaced by 1 for placed and 0 for unplaced

[7]:
```
data = df

status = {'Placed': 1,'Not Placed': 0}
data['status'] = [status[item] for item in data['status']]
```

[8]:
```
data.head()
```

[8]:
```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0      1      M  67.00   Others  91.00   Others  Commerce     58.00
1      2      M  79.33  Central  78.33   Others   Science     77.48
2      3      M  65.00  Central  68.00  Central      Arts     64.00
3      4      M  56.00  Central  52.00  Central   Science     52.00
4      5      M  85.80  Central  73.60  Central  Commerce     73.30

   degree_t workex  etest_p specialisation  mba_p  status    salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80       1  270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28       1  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80       1  250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43       0       0.0
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50       1  425000.0
```

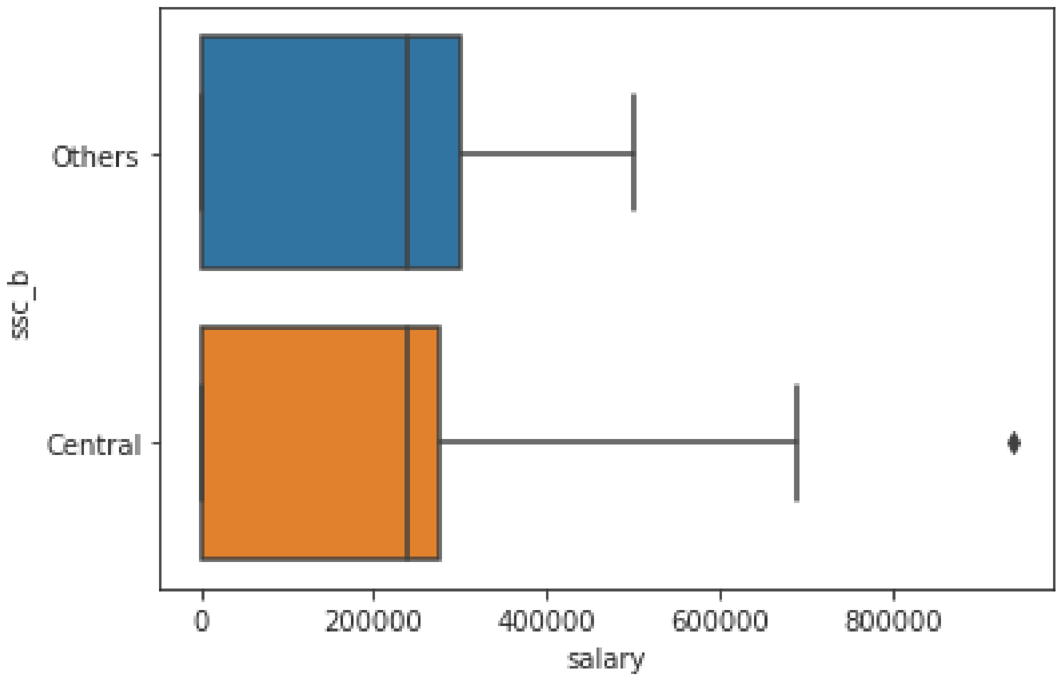
[9]:
```
df.describe()
```

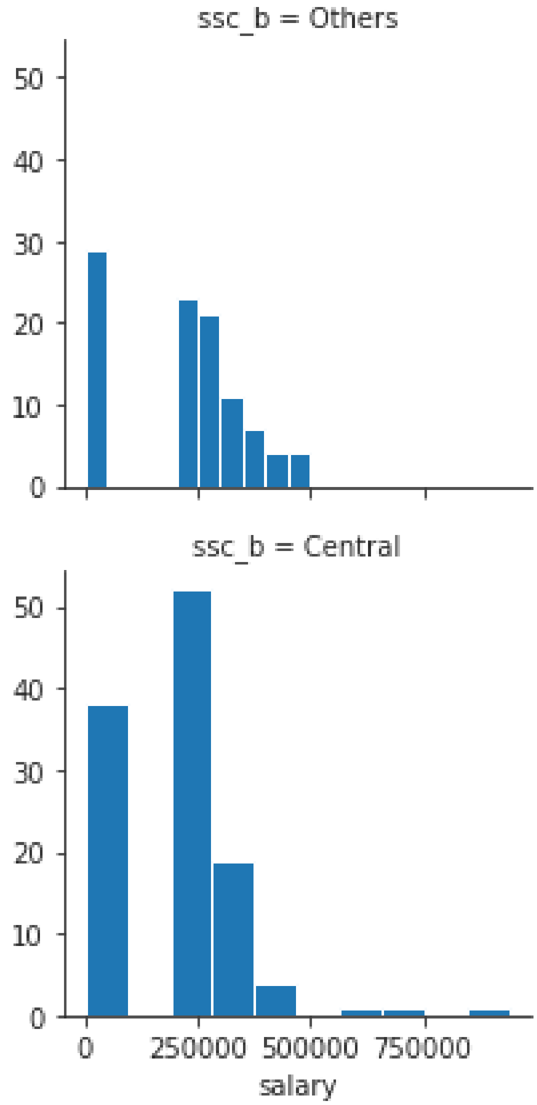
[9]:
```
            sl_no       ssc_p       hsc_p    degree_p     etest_p       mba_p  \
count  215.000000  215.000000  215.000000  215.000000  215.000000  215.000000
mean   108.000000   67.303395   66.333163   66.370186   72.100558   62.278186
std     62.209324   10.827205   10.897509    7.358743   13.275956    5.833385
min      1.000000   40.890000   37.000000   50.000000   50.000000   51.210000
25%     54.500000   60.600000   60.900000   61.000000   60.000000   57.945000
50%    108.000000   67.000000   65.000000   66.000000   71.000000   62.000000
75%    161.500000   75.700000   73.000000   72.000000   83.500000   66.255000
max    215.000000   89.400000   97.700000   91.000000   98.000000   77.890000

           status         salary
count  215.000000     215.000000
mean     0.688372  198702.325581
std      0.464240  154780.926716
min      0.000000       0.000000
25%      0.000000       0.000000
50%      1.000000  240000.000000
75%      1.000000  282500.000000
```

```
max      1.000000   940000.000000
```

[10]:
```
def plot(data,x,y):
    plt.Figure(figsize =(10,10))
    sns.boxplot(x = data[x],y= data[y])
    g = sns.FacetGrid(data, row = y)
    g = g.map(plt.hist,x)
    plt.show()
```

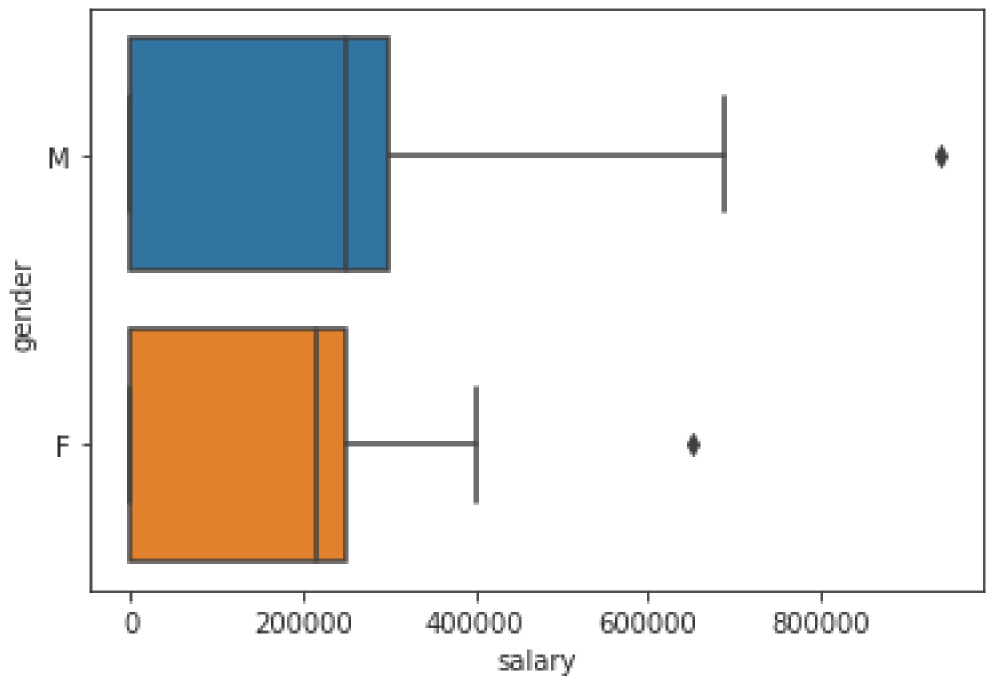
[11]:
```
sns.set_style("ticks")
plot(data,"salary","ssc_b")
```

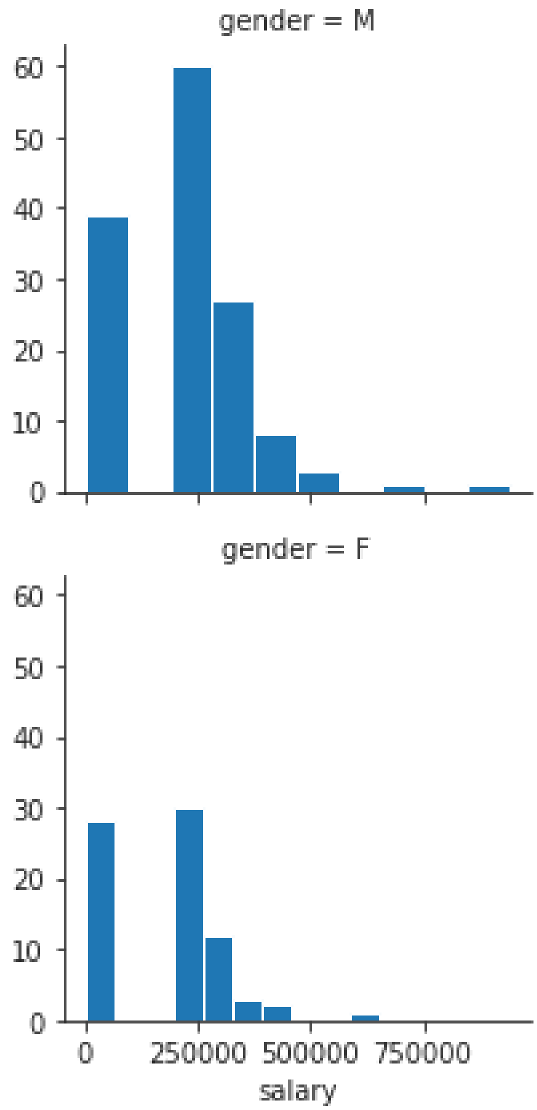




Although the median salary for both central and state board remains same, the highest package given is higher for students of central board. This happens because co-curricular activities provided to students are far better than those of state board. Prominently the stage fear and hesitation cause is removed for students coming from central board schools. From the histogram, it is seen that unplaced students are more for central schools.However overall placement stats remains same, thus choice of board for class 10th doesn't affects much for the placement.

[12]:
```
sns.set_style("ticks")
plot(data, "salary", "gender")
```

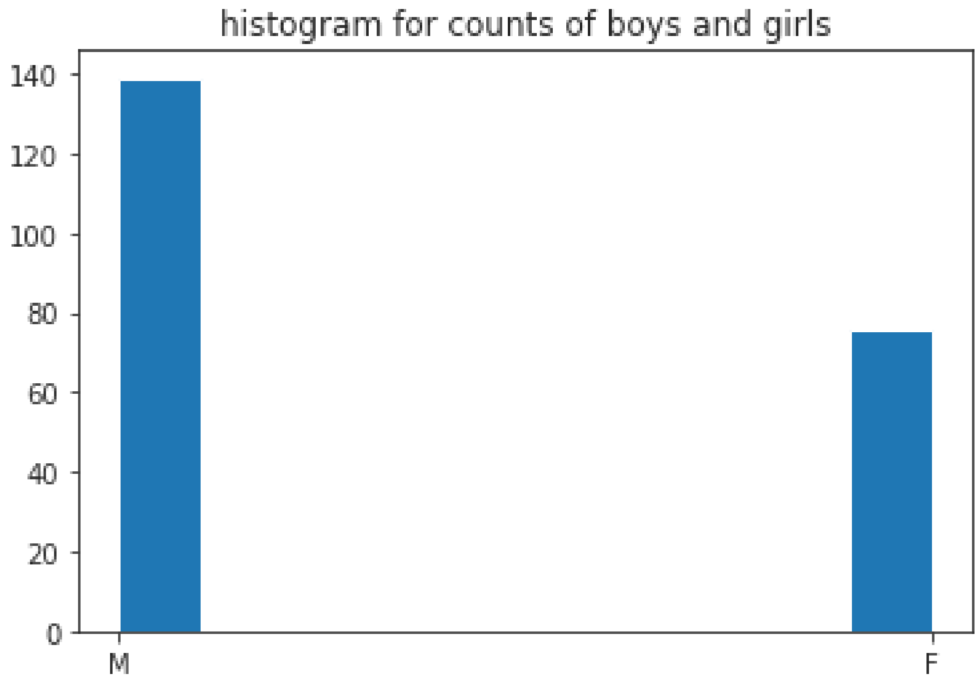

M
F
gender
0
200000
400000
600000
800000
salary



reference code for above plots

[13]:
```
plt.hist(data['gender'])
plt.title("histogram for counts of boys and girls")
```

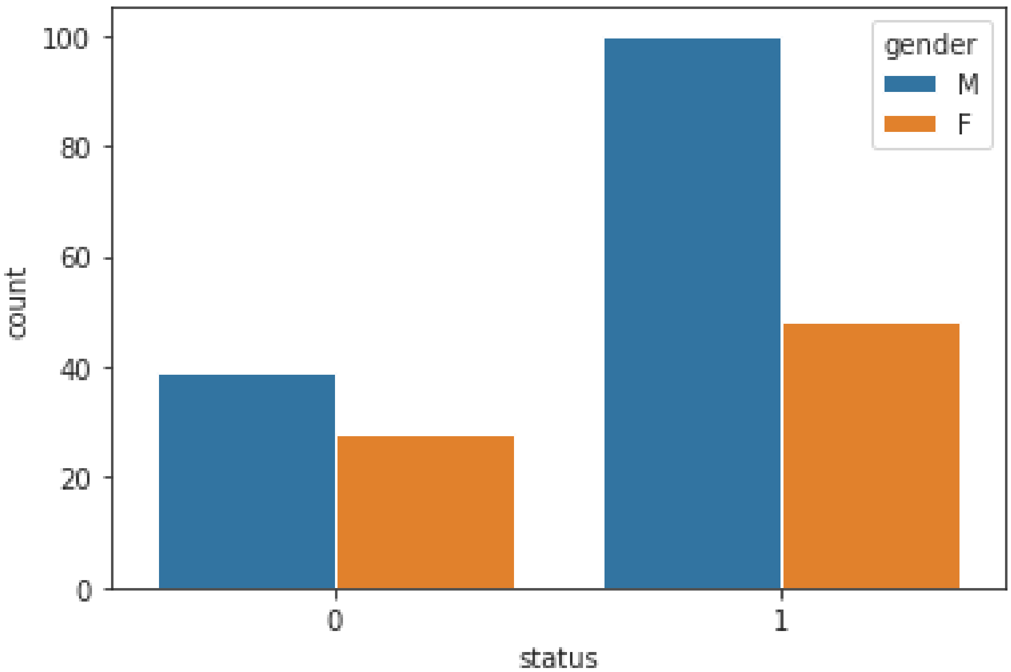
[13]: Text(0.5, 1.0, 'histogram for counts of boys and girls')



As from the boxplot we can see median salary for boys is greater than girls and so is the highest package. From the histogram, the number of unplaced girls and boys are 30 and 40 respectively. Althoug the number of unplaed girls seems less, but compared to the strngth of males and females, it is relatively high.

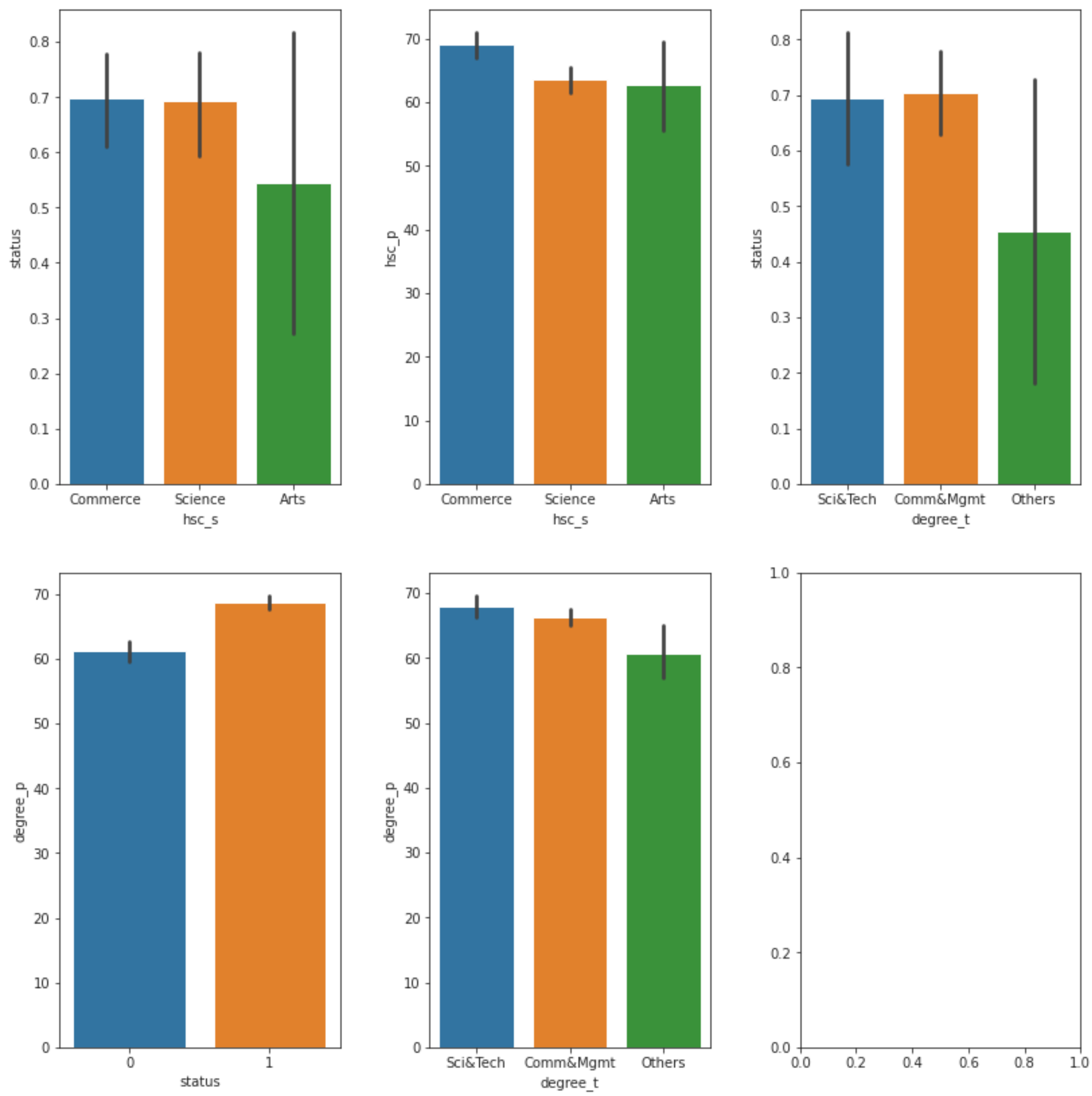
[14]:
```
sns.countplot(data['status'],hue=data['gender'])
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff1df6a9e80>
```



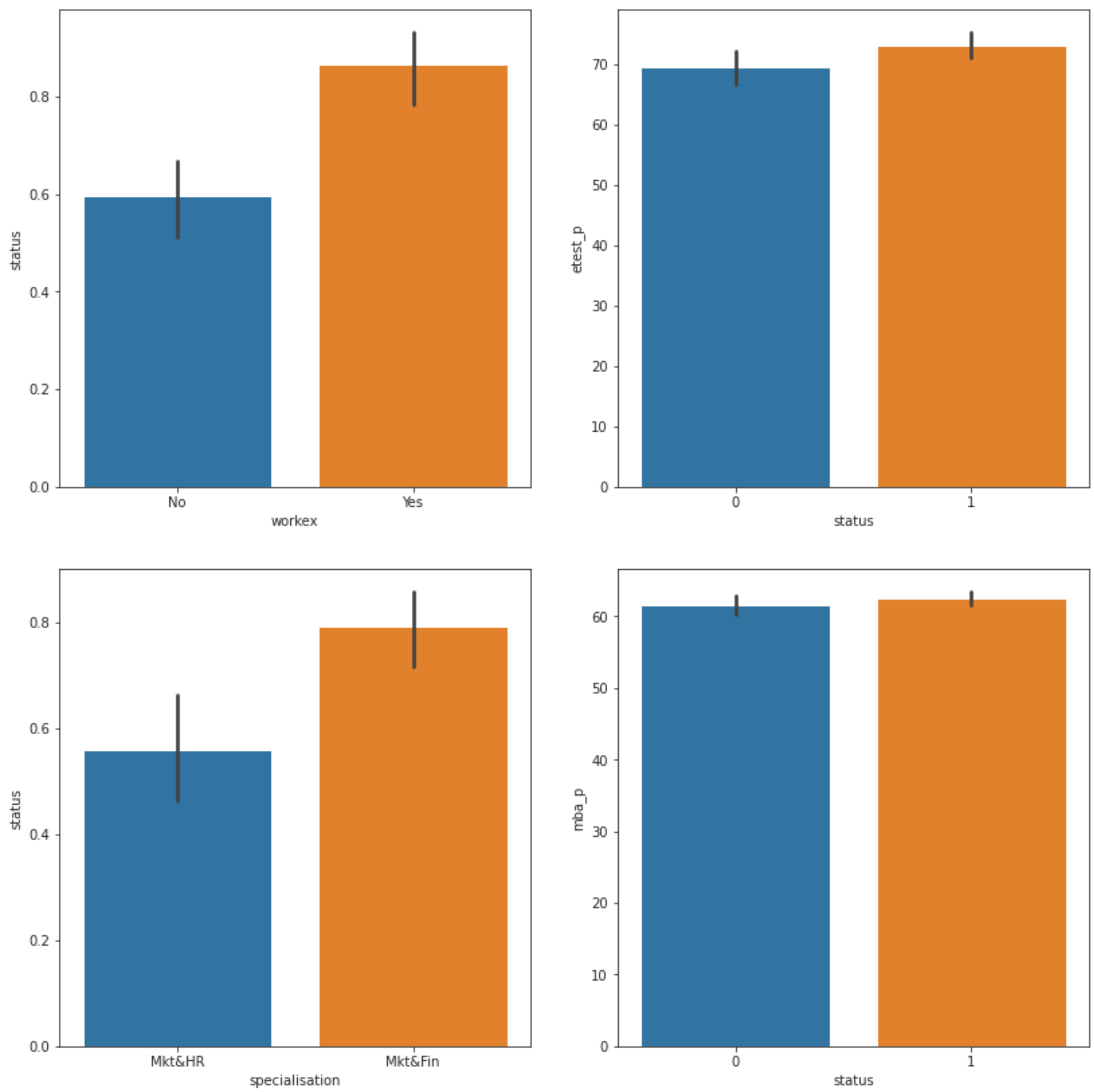
As we can see the number of placed students population of boys is higher than girls, although less number of girls did sit for the placement.This shows boys have slight better chance of getting hired than girls.

[15]:
```
fig, axes = plt.subplots(2,3, figsize=(12,12))
sns.barplot(x="hsc_s", y="status", data=data, ax = axes[(0,0)] )
sns.barplot(x="hsc_s", y="hsc_p", data=data, ax = axes[(0,1)])
sns.barplot(x="degree_t", y="status", data=data, ax = axes[(0,2)])
sns.barplot(x="status", y="degree_p", data=data, ax = axes[(1,0)])
sns.barplot(x="degree_t", y="degree_p", data=data, ax = axes[(1,1)])
plt.tight_layout(pad = 3)
```



From above plots it is clear that number of jobs available for commerce and science students is greater than that of students for other streams. Also it can be seen, the scores of class 12th do effect abit, although the impact is not that much high. Also the score of degree affects the placement, however maintaining an average score is enough to land in good jobs.

[16]:
```
fig, axes = plt.subplots(2,2, figsize=(12,12))
sns.barplot(x="workex", y="status", data=data, ax = axes[(0,0)])
sns.barplot(x="status", y="etest_p", data=data, ax = axes[(0,1)])
sns.barplot(x="specialisation", y="status", data=data, ax = axes[(1,0)])
sns.barplot(x="status", y="mba_p", data=data, ax = axes[(1,1)])
plt.tight_layout(pad = 3)
```



So, the first graph gives clear understanding that a work experience much preferable for getting into a good job,so one must go for good internships during college days. The employement test scores, doesn't matter much so the scores of mba, however a decent average score must be maintained. The marketing and finance student have more job opprtunities than H.R. for mba degree.

4.0.1 Thus the key factor would be for getting placed is earning an internship. Mostly the scores are discarded but they actually have some importance for the placement. Also chosing the stream puts huge impact for future growth of carrier

5 Data Preprocessing and Feature Engineering

1. Backward difference encoding of the categorical features and features containing texts.
2. Dropping the features not relevant or which puts less impact on the model.

3. Extracting important features if needed.
4. Using PCA/t-SNE for distribution visualization (if needed).
5. Standardizing the dataset before training.

[17]:
```
data.head()
```

[17]:
```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0      1      M  67.00   Others  91.00   Others  Commerce     58.00
1      2      M  79.33  Central  78.33   Others   Science     77.48
2      3      M  65.00  Central  68.00  Central      Arts     64.00
3      4      M  56.00  Central  52.00  Central   Science     52.00
4      5      M  85.80  Central  73.60  Central  Commerce     73.30

   degree_t workex  etest_p specialisation  mba_p  status    salary
0  Sci&Tech     No     55.0         Mkt&HR  58.80       1  270000.0
1  Sci&Tech    Yes     86.5        Mkt&Fin  66.28       1  200000.0
2 Comm&Mgmt     No     75.0        Mkt&Fin  57.80       1  250000.0
3  Sci&Tech     No     66.0         Mkt&HR  59.43       0       0.0
4 Comm&Mgmt     No     96.8        Mkt&Fin  55.50       1  425000.0
```

5.1 Encoding the data

[18]:
```
# encoding for the features

import category_encoders as ce
encoder = ce.BackwardDifferenceEncoder(cols=['ssc_b', "hsc_b", "hsc_s",␣
↪"degree_t", "workex", "specialisation", "gender"])
data_new = encoder.fit_transform(data)

data_new.head()
```

[18]:
```
   intercept  sl_no  gender_0  ssc_p  ssc_b_0  hsc_p  hsc_b_0   hsc_s_0  \
0          1      1      -0.5  67.00     -0.5  91.00     -0.5 -0.666667
1          1      2      -0.5  79.33      0.5  78.33     -0.5  0.333333
2          1      3      -0.5  65.00      0.5  68.00      0.5  0.333333
3          1      4      -0.5  56.00      0.5  52.00      0.5  0.333333
4          1      5      -0.5  85.80      0.5  73.60      0.5 -0.666667

    hsc_s_1  degree_p  degree_t_0  degree_t_1  workex_0  etest_p  \
0 -0.333333     58.00   -0.666667   -0.333333      -0.5     55.0
1 -0.333333     77.48   -0.666667   -0.333333       0.5     86.5
2  0.666667     64.00    0.333333   -0.333333      -0.5     75.0
3 -0.333333     52.00   -0.666667   -0.333333      -0.5     66.0
4 -0.333333     73.30    0.333333   -0.333333      -0.5     96.8

   specialisation_0  mba_p  status    salary
0              -0.5  58.80       1  270000.0
```

```
1                0.5  66.28       1  200000.0
2                0.5  57.80       1  250000.0
3               -0.5  59.43       0       0.0
4                0.5  55.50       1  425000.0
```

the code for backward difference encoding is taken from the mentioned source . In dataset as shown above we have encoded the string categorical data with some encoded values, so that it becomes relevant for models to learn. One hot encoding may suffer with the problem of curse of dimensionality, thus backward difference encoding is used.

5.2 Dropping unneccessary features

[19]:
```python
#dropping unnecessary features
data_new.drop(['intercept','sl_no'], axis=1, inplace=True)
```

[20]:
```python
data_new.head()
```

[20]:
```
   gender_0  ssc_p  ssc_b_0  hsc_p  hsc_b_0   hsc_s_0   hsc_s_1  degree_p  \
0      -0.5  67.00     -0.5  91.00     -0.5 -0.666667 -0.333333     58.00
1      -0.5  79.33      0.5  78.33     -0.5  0.333333 -0.333333     77.48
2      -0.5  65.00      0.5  68.00      0.5  0.333333  0.666667     64.00
3      -0.5  56.00      0.5  52.00      0.5  0.333333 -0.333333     52.00
4      -0.5  85.80      0.5  73.60      0.5 -0.666667 -0.333333     73.30

   degree_t_0  degree_t_1  workex_0  etest_p  specialisation_0  mba_p  status  \
0   -0.666667   -0.333333      -0.5     55.0              -0.5  58.80       1
1   -0.666667   -0.333333       0.5     86.5               0.5  66.28       1
2    0.333333   -0.333333      -0.5     75.0               0.5  57.80       1
3   -0.666667   -0.333333      -0.5     66.0              -0.5  59.43       0
4    0.333333   -0.333333      -0.5     96.8               0.5  55.50       1

     salary
0  270000.0
1  200000.0
2  250000.0
3       0.0
4  425000.0
```

5.3 Extracting important features

[21]:
```python
x = data_new["salary"]
```

[22]:
```python
labels = data_new['status']
features = data_new.iloc[:, :-2 ]
features = pd.concat([features, x], axis=1, join='inner')
features.head()
```

[22]:
```
   gender_0  ssc_p  ssc_b_0  hsc_p  hsc_b_0   hsc_s_0   hsc_s_1  degree_p  \
0      -0.5  67.00     -0.5  91.00     -0.5 -0.666667 -0.333333     58.00
1      -0.5  79.33      0.5  78.33     -0.5  0.333333 -0.333333     77.48
2      -0.5  65.00      0.5  68.00      0.5  0.333333  0.666667     64.00
3      -0.5  56.00      0.5  52.00      0.5  0.333333 -0.333333     52.00
4      -0.5  85.80      0.5  73.60      0.5 -0.666667 -0.333333     73.30

   degree_t_0  degree_t_1  workex_0  etest_p  specialisation_0  mba_p  \
0   -0.666667   -0.333333      -0.5     55.0              -0.5  58.80
1   -0.666667   -0.333333       0.5     86.5               0.5  66.28
2    0.333333   -0.333333      -0.5     75.0               0.5  57.80
3   -0.666667   -0.333333      -0.5     66.0              -0.5  59.43
4    0.333333   -0.333333      -0.5     96.8               0.5  55.50

     salary
0  270000.0
1  200000.0
2  250000.0
3       0.0
4  425000.0
```

[23]:
```python
labels.head()
```

[23]:
```
0    1
1    1
2    1
3    0
4    1
Name: status, dtype: int64
```

[24]:
```python
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

# feature extraction
model = LogisticRegression(solver='lbfgs')
rfe = RFE(model, 10)
fit = rfe.fit(features, labels)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False  True False  True False  True  True  True False  True
True  True
 False  True  True]
Feature Ranking: [2 1 6 1 5 1 1 1 3 1 1 1 4 1 1]
```

The important features have been extracted and is stored in X_selected. We will use these features for training our ML algorithm and further look over the results. I chose 10 features important for the training purpose, although one is free to chose of its own.The important features for placement stats are: **["ssc_p", "hsc_p", "hsc_s_0", "hsc_s_1", "degree_p", "degree_t_1", "workex_0", "etest_p", "mba_p", "salary"]** To have some insights about features we will use dimensionality reductin in below setion and have look hoe much is the separability of the features is present. The code snippet for feature selection is taken from https://machinelearningmastery.com/feature-selection-machine-learning-python/.

Some other sources from where help is taken for feature selection is https://www.datacamp.com/community/tutorials/feature-selection-python

We will keep only the 10 most important features for working further.

[25]:
```
features.drop(["gender_0", "ssc_b_0", "hsc_b_0", "degree_t_0",␣
 ↪"specialisation_0"], axis=1, inplace=True)
features.head()
```

[25]:
```
   ssc_p  hsc_p   hsc_s_0   hsc_s_1  degree_p  degree_t_1  workex_0  etest_p  \
0  67.00  91.00 -0.666667 -0.333333     58.00   -0.333333      -0.5     55.0
1  79.33  78.33  0.333333 -0.333333     77.48   -0.333333       0.5     86.5
2  65.00  68.00  0.333333  0.666667     64.00   -0.333333      -0.5     75.0
3  56.00  52.00  0.333333 -0.333333     52.00   -0.333333      -0.5     66.0
4  85.80  73.60 -0.666667 -0.333333     73.30   -0.333333      -0.5     96.8

   mba_p    salary
0  58.80  270000.0
1  66.28  200000.0
2  57.80  250000.0
3  59.43       0.0
4  55.50  425000.0
```

5.4 Applying t-SNE and using it for getting insight of features

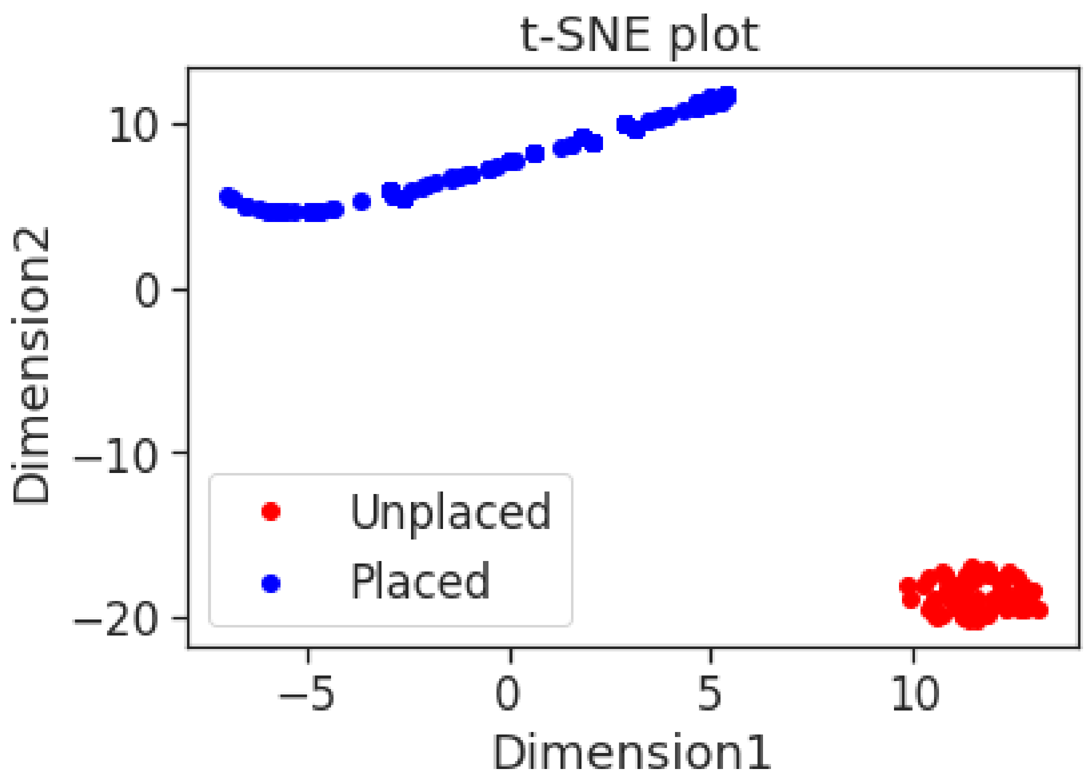
[26]:
```
df_final = features
df_final.to_numpy()
distinct_labels = list(set(labels))
distinct_labels
```

[26]:
```
[0, 1]
```

[27]:
```
from sklearn.manifold import TSNE
import matplotlib.patheffects as PathEffects

y = labels
X_raw = df_final
y_raw = np.array(y, dtype = 'int')
tsne = TSNE(n_components=2, random_state=0, perplexity = 50, n_iter = 5000)
X_2d = tsne.fit_transform(X_raw)
```

```
X1 = X_2d[:,0:1]
Y1 = X_2d[:,1:2]

sns.set_style('ticks')
sns.set_palette('muted')
sns.set_context("notebook", font_scale=1.5,
                rc={"lines.linewidth": 2.5})


category_to_color = {0: 'red', 1: 'blue'}
category_to_label = {0: 'Unplaced', 1:"Placed"}

fig, ax = plt.subplots(1,1)
for category, color in category_to_color.items():
    mask = y == category
    ax.plot(X_2d[mask, 0], X_2d[mask, 1], 'o',
            color=color, label=category_to_label[category], ms = 6)

ax.legend(loc='best')
ax.axis('on')
ax.axis('tight')
plt.xlabel('Dimension1')
plt.ylabel('Dimension2')
plt.title(' t-SNE plot')
```

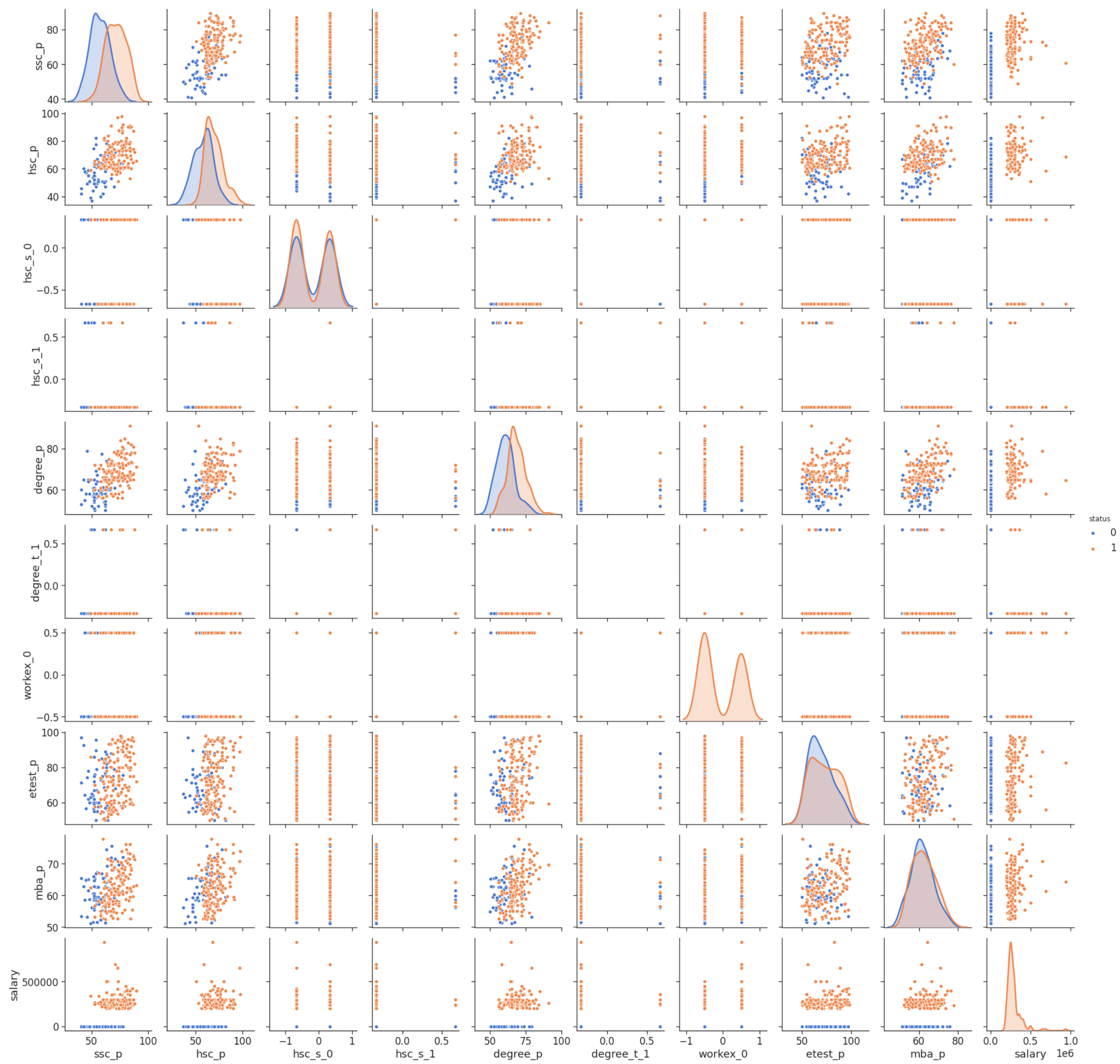
[27]:
```
Text(0.5, 1.0, ' t-SNE plot')
```



Looking at the above t-SNE plot it is very easy to conclude that features are separable, also the two classes are shown. Now we will move further to train models and do predictive analysis and verify models by using different metrices for our classifiers or regressor models. Before this there is one last step of normalizing the dataset so that all the values lies in a particular range for the robustness of our model.Also we will look at the pairplots of the features to have a look at little dependencies of features on each other.

[28]:
```
df_plot = pd.concat([features, labels], axis=1, join='inner')
sns.pairplot(df_plot, vars=df_plot.columns[:-1], hue = 'status')
```

[28]: <seaborn.axisgrid.PairGrid at 0x7ff1dd4c4518>



5.5 Standardizing the data

[29]:
```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
x = scaler.fit_transform(X_raw)
y = y_raw
```

6 Training different classifiers and measuring the accuracy values

[43]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
```

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.18)
```

[31]:
```python
model1 = RandomForestClassifier()
model1.fit(x_train,y_train)
model1.score(x_test,y_test)
predictions1 = model1.predict(x_test)
print(confusion_matrix(y_test,predictions1))
print(classification_report(y_test,predictions1))
```

```
[[14  0]
 [ 0 25]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        25

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39
```

[32]:
```python
model2 = DecisionTreeClassifier()
model2.fit(x_train,y_train)

predictions2 = model2.predict(x_test)
print(confusion_matrix(y_test,predictions2))
print(classification_report(y_test,predictions2))
```

```
[[14  0]
 [ 0 25]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        25

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39
```

[33]:
```python
model3 = KNeighborsClassifier()
model3.fit(x_train,y_train)
```

```
predictions3 = model3.predict(x_test)
print(confusion_matrix(y_test,predictions3))
print(classification_report(y_test,predictions3))
```

```
[[14  0]
 [ 0 25]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        25

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39
```

[34]:
```
model4 = XGBClassifier()
model4.fit(x_train,y_train)
predictions4 = model4.predict(x_test)
print(confusion_matrix(y_test,predictions4))
print(classification_report(y_test,predictions4))
```

```
[[14  0]
 [ 0 25]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        25

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39
```

[35]:
```
model5 = GaussianNB()
model5.fit(x_train,y_train)
predictions5 = model5.predict(x_test)
print(confusion_matrix(y_test,predictions5))
print(classification_report(y_test,predictions5))
```

```
[[14  0]
 [ 0 25]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        25
```

```
    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39
```

[44]:
```
#neural network classifier
model6 = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2),␣
 ↪random_state=1)
model6.fit(x_train,y_train)
predictions6 = model6.predict(x_test)
print(confusion_matrix(y_test,predictions6))
print(classification_report(y_test,predictions6))
```

```
[[17  0]
 [ 0 22]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        22

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39
```

Looking at the confusion matrix, it appears that all the models are performing exceptionally well. Some reasons which I can think of is: - 1. Since addressing the case of placed and unplaced becomes binary classification problem, so most of the models are able to perform well as the dataset size is too small. 2. The t-SNE plot shows that both the classes are ar too apart and thus the features are searable, due to which complexity has almost gone down. 3. Feature selection step extracts only the most important features, due to which the data becomes quite clean for training purpose. 4. Also the presence of outlier is not there or is minimal as we can see from the plots above.

7 Regression for predicting salary

After looking at the classification model, we are assure that models are performing well for this binary classification task for predicting whether soeone will be placed or not, however let's have a different look at the problem and try to predict the salary a student would recieve depending on the important features we have. in this we will use Ensemble models for regression purpose and analyze the prformance of the models.

[36]:
```
#preparing the data for regression
data_reg =features
data_reg.head()
```

[36]:

```
   ssc_p  hsc_p   hsc_s_0   hsc_s_1  degree_p  degree_t_1  workex_0  etest_p  \
0  67.00  91.00 -0.666667 -0.333333     58.00   -0.333333      -0.5     55.0
1  79.33  78.33  0.333333 -0.333333     77.48   -0.333333       0.5     86.5
2  65.00  68.00  0.333333  0.666667     64.00   -0.333333      -0.5     75.0
3  56.00  52.00  0.333333 -0.333333     52.00   -0.333333      -0.5     66.0
4  85.80  73.60 -0.666667 -0.333333     73.30   -0.333333      -0.5     96.8

   mba_p    salary
0  58.80  270000.0
1  66.28  200000.0
2  57.80  250000.0
3  59.43       0.0
4  55.50  425000.0
```

[39]:

```
x_feat = data_reg.iloc[:,:-1]
y_target = data_reg['salary']
x_reg = x_feat
x_reg.to_numpy()
y_reg = np.array(y_target, dtype = 'int')
scale = preprocessing.StandardScaler()
x_reg = scale.fit_transform(x_reg)
```

[40]:

```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import cross_val_score
x_train, x_test, y_train, y_test = train_test_split(x_reg, y_reg, test_size = 0.
 ↪18)
```

[41]:

```
reg1 = XGBRegressor(learning_rate=0.01, n_estimators=2000)
reg2 = RandomForestRegressor(n_estimators=2000)
reg1_scores = cross_val_score(reg1, x_train, y_train, cv=10)
reg2_scores = cross_val_score(reg2, x_train, y_train, cv=10)
print("XGB Regression: ",np.mean(reg1_scores))
print("Random Forest Regression: ",np.mean(reg2_scores))
```

```
XGB Regression:  -0.07134193327738116
Random Forest Regression:  0.17928660719525036
```

Looking at the cross validation scores, it appears that models are not performing well, however we will move further with predictions to look at the actual results of our models.

[42]:

```
#predicting

reg1.fit(x_train, y_train)
pred_1 = reg1.predict(x_test)
print(reg1.score(x_test, y_test))
```

```
reg2.fit(x_train, y_train)
pred_2 = reg2.predict(x_test)
print(reg2.score(x_test, y_test))
```

```
0.5295631234476423
0.5668521159898918
```

The R^2 scores seems to be low and thus our models are not performing upto the mark for predicting the values.

[41]:
```
df1_pred = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': pred_1.
 ↪flatten()})
df1_pred
```

[41]:

	Actual	Predicted
0	0	249989.968750
1	0	58436.476562
2	420000	300524.468750
3	300000	352648.593750
4	0	195.830399
5	265000	254500.609375
6	265000	175795.546875
7	275000	299030.187500
8	0	200924.687500
9	360000	280124.656250
10	230000	230919.890625
11	0	182551.921875
12	220000	152744.437500
13	265000	319764.437500
14	225000	269825.656250
15	240000	235976.921875
16	0	173089.515625
17	0	-1599.704102
18	350000	117167.507812
19	200000	232988.953125
20	280000	314881.656250
21	260000	16897.904297
22	650000	278829.218750
23	260000	289719.937500
24	500000	76409.164062
25	0	-4225.056152
26	270000	382697.937500
27	287000	263694.593750
28	250000	305036.906250
29	0	-45853.691406
30	300000	248491.828125
31	275000	206242.890625
32	340000	9768.720703

```
33  250000  257562.125000
34  240000  158333.265625
35  231000  295746.218750
36       0   33046.054688
37  200000  294049.562500
38  240000  189361.906250
```

[42]:
```python
df2_pred = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': pred_2.
 ↪flatten()})
df2_pred
```

[42]:

	Actual	Predicted
0	0	225168.0
1	0	38818.0
2	420000	277648.0
3	300000	288573.0
4	0	500.0
5	265000	245404.0
6	265000	215964.0
7	275000	275659.0
8	0	117515.0
9	360000	270106.0
10	230000	246599.0
11	0	246495.0
12	220000	110097.0
13	265000	273449.0
14	225000	252497.0
15	240000	268174.0
16	0	223192.0
17	0	21031.0
18	350000	150612.0
19	200000	256067.0
20	280000	297138.0
21	260000	17933.0
22	650000	262763.0
23	260000	275989.0
24	500000	152468.0
25	0	775.0
26	270000	272019.0
27	287000	262581.0
28	250000	283213.0
29	0	8925.0
30	300000	192496.0
31	275000	223110.0
32	340000	8125.0
33	250000	257611.0
34	240000	200635.0

```
35  231000   266761.0
36       0    38485.0
37  200000   311200.0
38  240000   236465.0
```

Looking at actual and predicted values, it appears that the result is fairly bad, as the students who are not placed, receive decent amount according to predicted result. This may be a hypothetical case, when of'ourse the student is placed with all those credentials, and would earn a nearby same salary. However the results are not consistent with the dataset, and thus we cannot rely on machine learning model to actually assume or expect some salary figures from the recruiters. This concludes our analysis on the recruitment dataset, where we tried to address the problem of classification and regression and came to a conclusion.

8 Conclusion and Discussion

1. We can say that for classification model, the results are pretty well and if the student has some decent credentials in college and better 12th score, there might be chances of earning good salary job.
2. The gender is taken into factor for placement, however it is very less.
3. With regression analysis, it came to us that an out of the box thinking and extra effort is needed to earn a decent salary, not just the degree, and marks.
4. Co- curriculars do affects as it adds to the holistic personality of the student.
5. Most important point is work experience and internships do effects the placements and obviously one must go for it during college days.
6. Another key factor is stream choice as science and commerce have better jobs than arts, however only chosing a stream won't help to earn decent,rather one must take into account above mentioned factors.

[ ]:

```

```