Campus DT+RF+logggggggggggggg

October 3, 2023

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df=pd.read_csv('Placement_Data_Full_Class.csv')
df=df.drop(columns='sl_no',axis=1)
```

[3]:
```
df.head(10)
```

[3]:
```
  gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p   degree_t  \
0      M  67.00   Others  91.00   Others  Commerce     58.00   Sci&Tech
1      M  79.33  Central  78.33   Others   Science     77.48   Sci&Tech
2      M  65.00  Central  68.00  Central      Arts     64.00  Comm&Mgmt
3      M  56.00  Central  52.00  Central   Science     52.00   Sci&Tech
4      M  85.80  Central  73.60  Central  Commerce     73.30  Comm&Mgmt
5      M  55.00   Others  49.80   Others   Science     67.25   Sci&Tech
6      F  46.00   Others  49.20   Others  Commerce     79.00  Comm&Mgmt
7      M  82.00  Central  64.00  Central   Science     66.00   Sci&Tech
8      M  73.00  Central  79.00  Central  Commerce     72.00  Comm&Mgmt
9      M  58.00  Central  70.00  Central  Commerce     61.00  Comm&Mgmt

  workex  etest_p specialisation  mba_p      status    salary
0     No    55.00         Mkt&HR  58.80      Placed  270000.0
1    Yes    86.50        Mkt&Fin  66.28      Placed  200000.0
2     No    75.00        Mkt&Fin  57.80      Placed  250000.0
3     No    66.00         Mkt&HR  59.43  Not Placed       NaN
4     No    96.80        Mkt&Fin  55.50      Placed  425000.0
5    Yes    55.00        Mkt&Fin  51.58  Not Placed       NaN
6     No    74.28        Mkt&Fin  53.29  Not Placed       NaN
7    Yes    67.00        Mkt&Fin  62.14      Placed  252000.0
8     No    91.34        Mkt&Fin  61.29      Placed  231000.0
9     No    54.00        Mkt&Fin  52.21  Not Placed       NaN
```

1 1 Data Preprocessing

[4]:
```
df.shape
```

[4]:
```
(215, 14)
```

[5]:
```
df.select_dtypes(include='object').nunique()
```

[5]:
```
gender            2
ssc_b             2
hsc_b             2
hsc_s             3
degree_t          3
workex            2
specialisation    2
status            2
dtype: int64
```

[6]:
```
df.isna().sum()
```

[6]:
```
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
```

[7]:
```
df.isna().mean()
```

[7]:
```
gender            0.000000
ssc_p             0.000000
ssc_b             0.000000
hsc_p             0.000000
hsc_b             0.000000
hsc_s             0.000000
degree_p          0.000000
degree_t          0.000000
workex            0.000000
etest_p           0.000000
```

```
specialisation    0.000000
mba_p             0.000000
status            0.000000
salary            0.311628
dtype: float64
```

[8]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   gender          215 non-null    object
 1   ssc_p           215 non-null    float64
 2   ssc_b           215 non-null    object
 3   hsc_p           215 non-null    float64
 4   hsc_b           215 non-null    object
 5   hsc_s           215 non-null    object
 6   degree_p        215 non-null    float64
 7   degree_t        215 non-null    object
 8   workex          215 non-null    object
 9   etest_p         215 non-null    float64
 10  specialisation  215 non-null    object
 11  mba_p           215 non-null    float64
 12  status          215 non-null    object
 13  salary          148 non-null    float64
dtypes: float64(6), object(8)
memory usage: 23.6+ KB
```

2 EDA

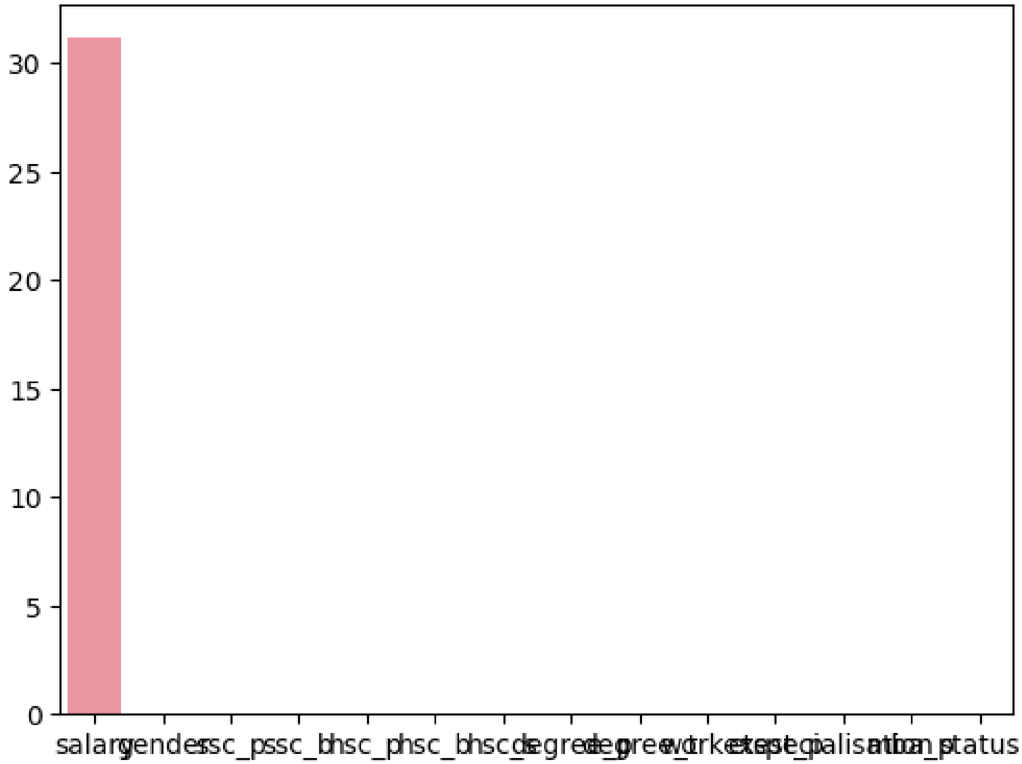
[9]:
```python
dfcat=df.select_dtypes(include='object')
```

[10]:
```python
cat_col=df.select_dtypes(include='object').columns
```

[11]:
```python
j=(df.isna().mean()*100).sort_values(ascending=False)
```

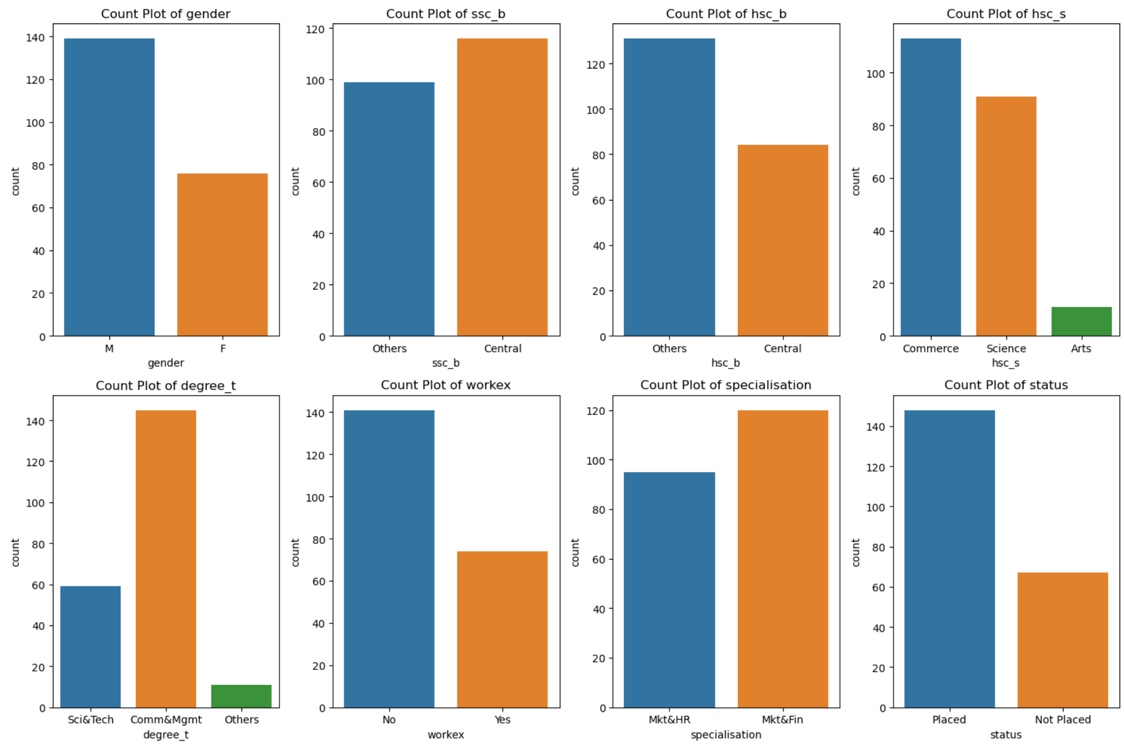
[12]:
```python
sns.barplot(x=j.index,y=j.values)
```

[12]: <Axes: >

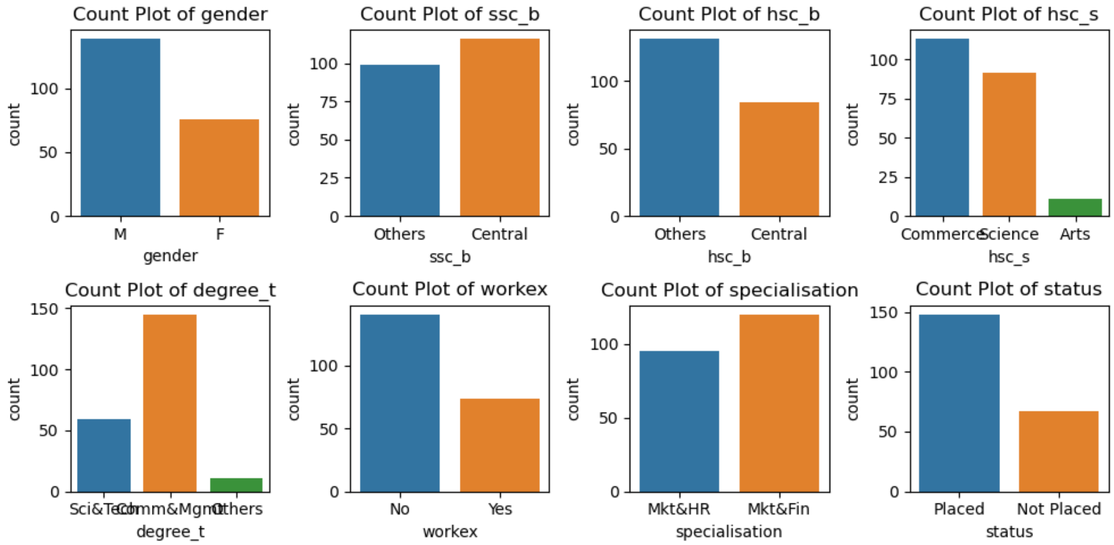


[13]:
```
def fun(col):
    fig, ax = plt.subplots(figsize=(7, 5))

    sns.countplot(data=df, x=col, ax=ax)
    ax.set_title(f'Count Plot of {col}')

    plt.tight_layout()
    plt.close()  # Close the figure to avoid displaying the individual plots␣
 ↪immediately

# Create two figures side by side (1 row and 2 columns)
fig, axes = plt.subplots(2, 4, figsize=(15, 10))

# Call the function for each categorical column in 'cat_col' and plot it in the␣
 ↪respective figure
for i, col in enumerate(cat_col):
    row_index = i // 4
    col_index = i % 4
    fun(col)
    sns.countplot(data=df, x=col, ax=axes[row_index, col_index])
    axes[row_index, col_index].set_title(f'Count Plot of {col}')
```

```
plt.tight_layout()
plt.show()
```



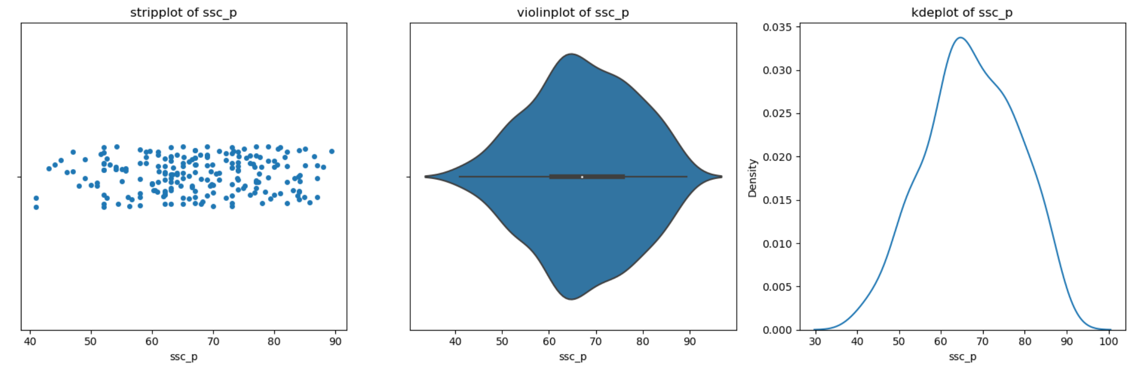
[14]:
```
#Categorical column bar plot
fig, axes = plt.subplots(2, 4, figsize=(10, 5))

# Call the countplot for each categorical column in 'cat_col' and plot it in the respective figure
for i, col in enumerate(cat_col):
    row_index = i // 4
    col_index = i % 4
    sns.countplot(data=df, x=col, ax=axes[row_index, col_index])
    axes[row_index, col_index].set_title(f'Count Plot of {col}')

plt.tight_layout()
plt.show()
```

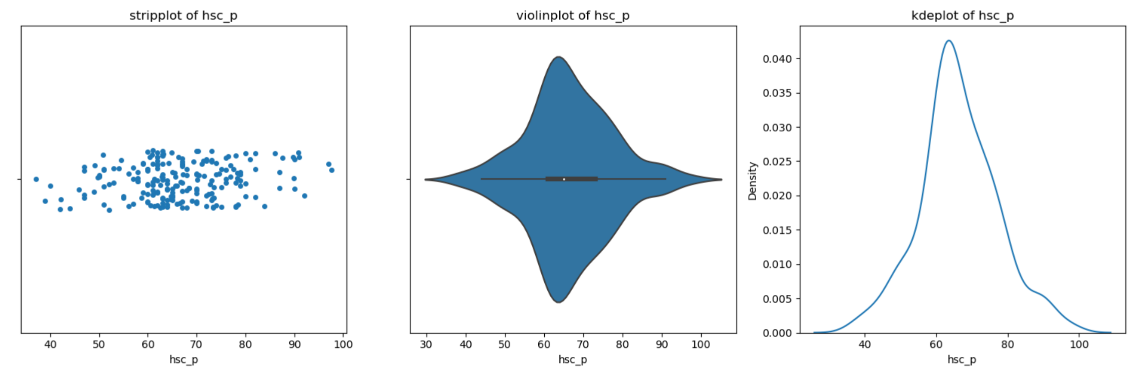


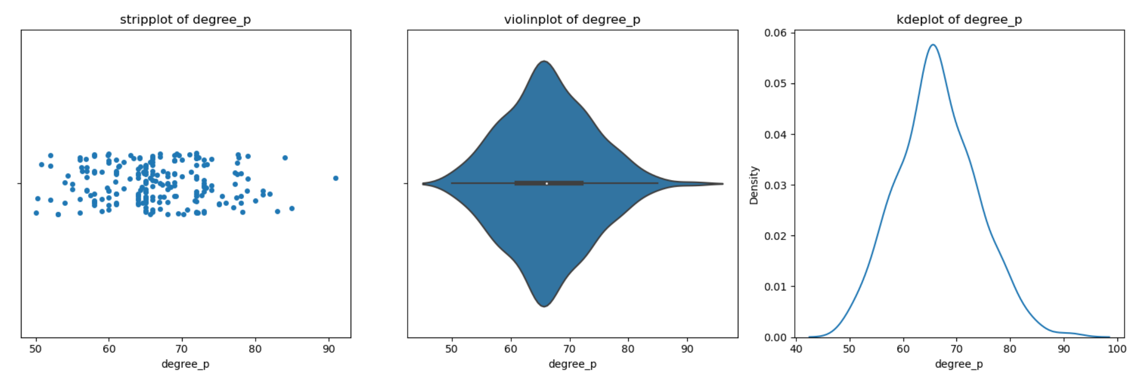
[15]:
```
dfnum=df.drop(dfcat.columns,axis=1)
```

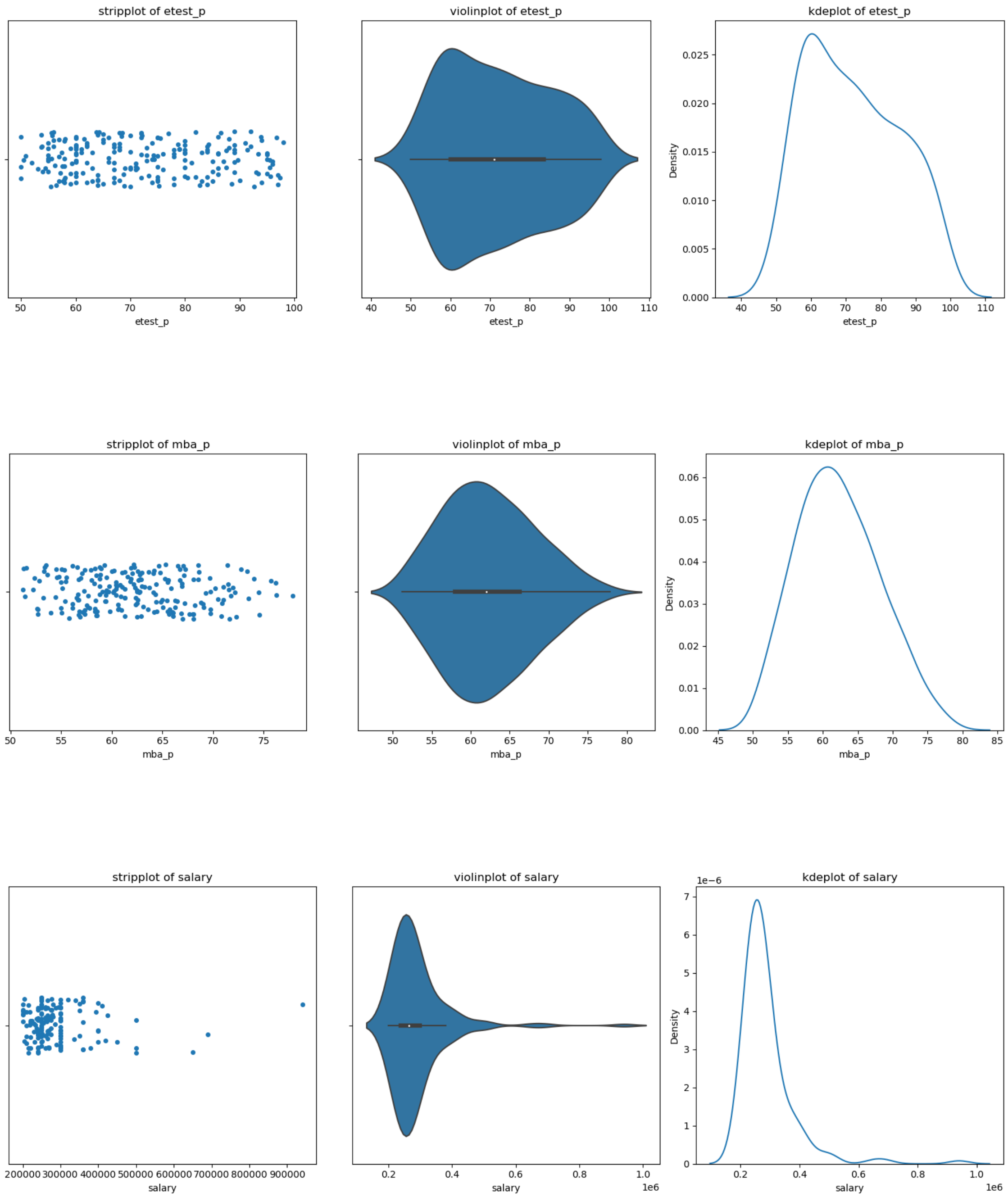
[16]:
```
dfnum
num_col=dfnum.columns
```

[17]:
```
#numerica; columns distribution
def numfunc(col):
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))

    plots = [
        ('stripplot', sns.stripplot),
        ('violinplot', sns.violinplot),
        ('kdeplot', sns.kdeplot)
    ]

    for i, (plot_title, plot_func) in enumerate(plots):
        plot_func(data=dfnum, x=col, ax=axes[i])
        axes[i].set_title(f'{plot_title} of {col}')

    plt.tight_layout()
    plt.show()

# Call the function for each numerical column in the DataFrame 'DFNUM'
for col in dfnum.columns:
    numfunc(col)
```


stripplot of ssc_p
ssc_p
violinplot of ssc_p
ssc_p
kdeplot of ssc_p
Density
ssc_p


stripplot of hsc_p
hsc_p
violinplot of hsc_p
hsc_p
kdeplot of hsc_p
Density
hsc_p

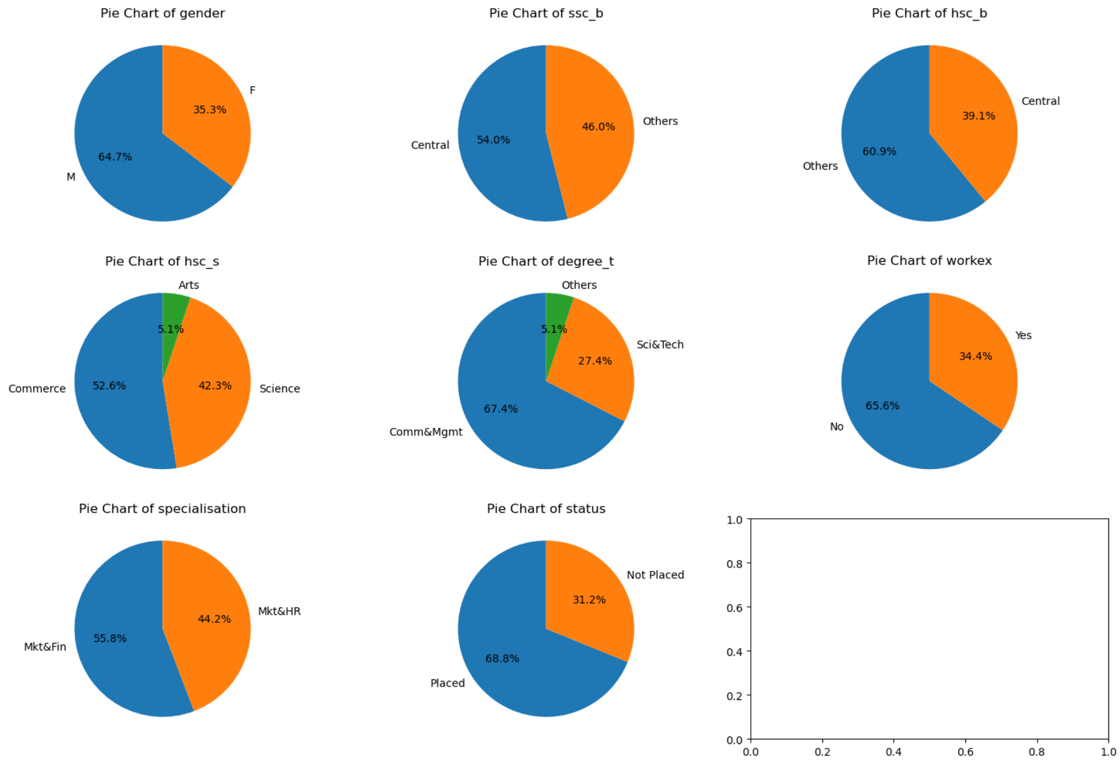

stripplot of degree_p
degree_p
violinplot of degree_p
degree_p
kdeplot of degree_p
Density
degree_p



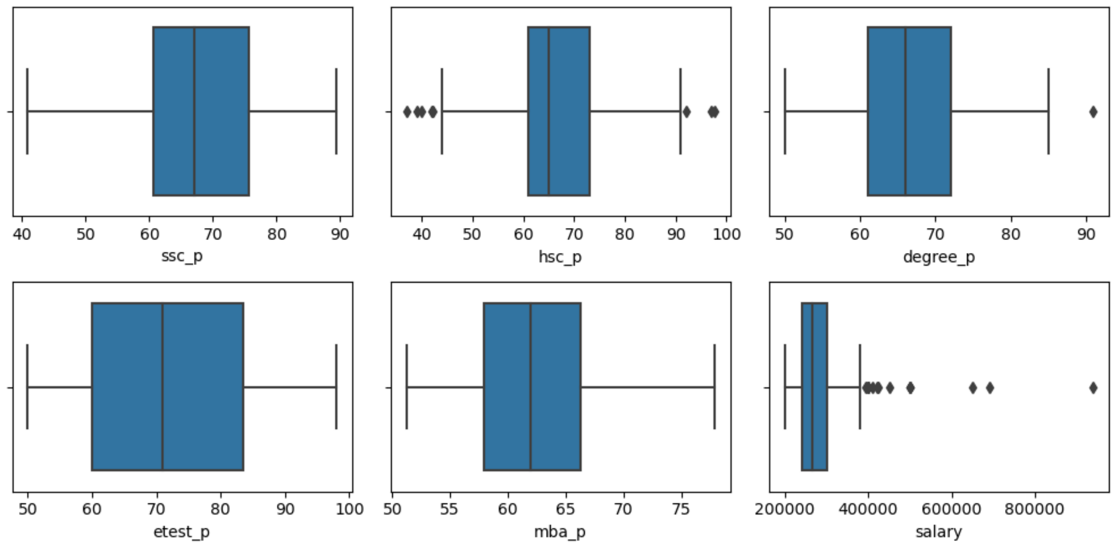
[18]:
```
def funccatcol_pie(df,cat_col):
    fig,axes =plt.subplots(3,3,figsize=(15,10))

    for i,col in enumerate(cat_col):
        r=i//3
        c=i%3
        pp=df[col].value_counts()
```

```
        axes[r,c].pie(pp.values,labels=pp.index,autopct='%1.1f%%', startangle=90, textprops={'fontsize': 10})
        axes[r,c].set_title(f'Pie Chart of {col}')
    plt.tight_layout()
    plt.show()
```

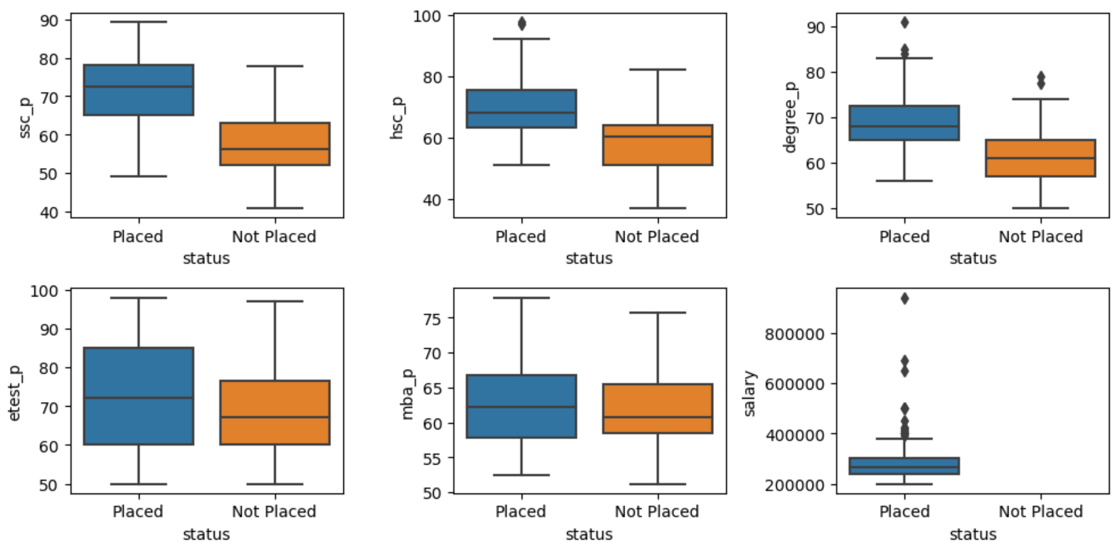
[19]:
```
funccatcol_pie(df,cat_col)
```



[20]:
```
def fff(dfnum,num_col):
    fig,axs=plt.subplots(nrows=2,ncols=3,figsize=(10,5))
    axs=axs.flatten()
    for i,col in enumerate(num_col):
        sns.boxplot(x=col,data=dfnum,ax=axs[i])
    fig.tight_layout()
    plt.show()

fff(dfnum,num_col)
```

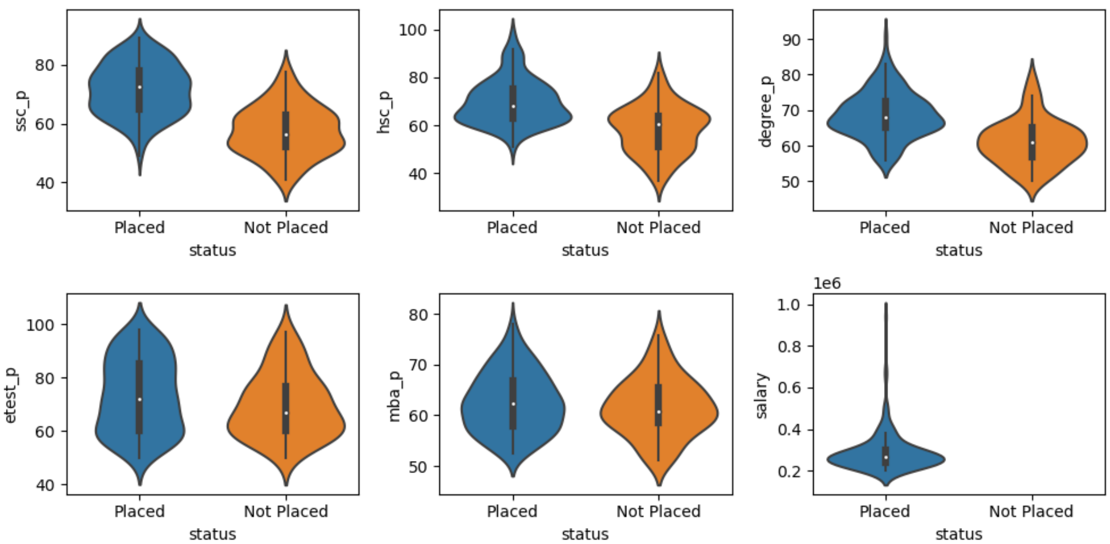


[21]:
```
def fff(df,num_col):
    fig,axs=plt.subplots(nrows=2,ncols=3,figsize=(10,5))
    axs=axs.flatten()
    for i,col in enumerate(num_col):
        sns.boxplot(y=col,x='status',data=df,ax=axs[i])
    fig.tight_layout()
    plt.show()

fff(df,num_col)
```



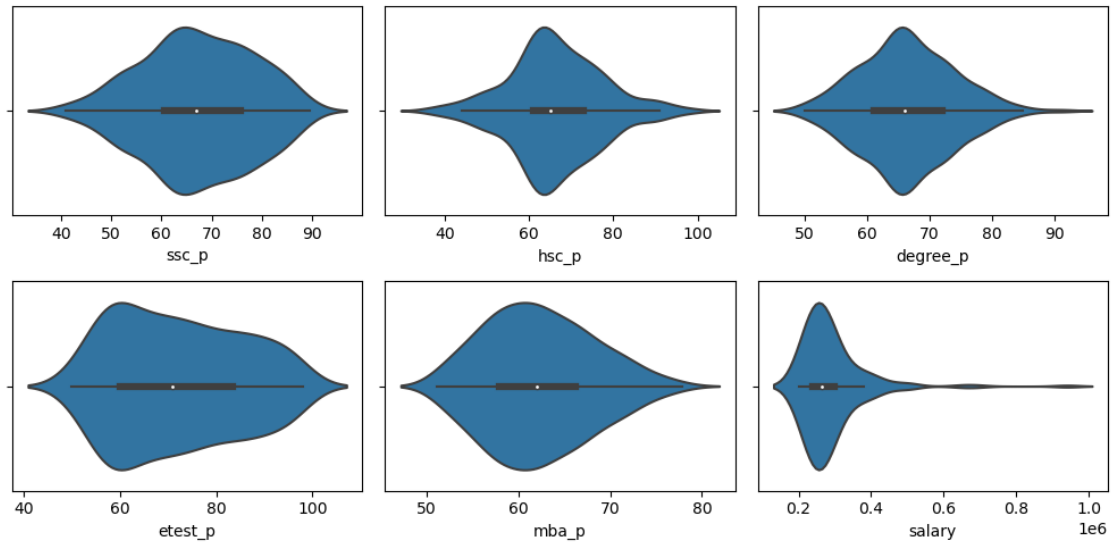
[22]:

```
def fff(df,num_col):
    fig,axs=plt.subplots(nrows=2,ncols=3,figsize=(10,5))
    axs=axs.flatten()
    for i,col in enumerate(num_col):
        sns.violinplot(y=col,x='status',data=df,ax=axs[i])
    fig.tight_layout()
    plt.show()

fff(df,num_col)
```



[23]:

```
def fff(df,num_col):
    fig,axs=plt.subplots(nrows=2,ncols=3,figsize=(10,5))
    axs=axs.flatten()
    for i,col in enumerate(num_col):
        sns.violinplot(x=col,data=dfnum,ax=axs[i])
    fig.tight_layout()
    plt.show()

fff(df,num_col)
```



3 2 Data Preprocessing

[24]:
```
empty_col=df.isna().mean()*100
```

[25]:
```
empty_col[empty_col>0]
```

[25]:
```
salary    31.162791
dtype: float64
```

[26]:
```
df['salary']=df['salary'].fillna(df['salary'].median())
```

[27]:
```
empty_col[empty_col>0]
```

[27]:
```
salary    31.162791
dtype: float64
```

4 Categoury in each categorical column

[28]:
```
for i in cat_col:
    print(f'{i}:{df[i].unique()}')
```

```
gender:['M' 'F']
ssc_b:['Others' 'Central']
hsc_b:['Others' 'Central']
hsc_s:['Commerce' 'Science' 'Arts']
degree_t:['Sci&Tech' 'Comm&Mgmt' 'Others']
workex:['No' 'Yes']
```

```
specialisation:['Mkt&HR' 'Mkt&Fin']
status:['Placed' 'Not Placed']
```

5 Label Encoding

[29]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
dfcat1=pd.DataFrame()
for col in df.select_dtypes(include=['object']).columns:
    label_encoder = LabelEncoder()
    dfcat1[col] = label_encoder.fit_transform(dfcat[col])
    print(f'{col}:{dfcat1[col].unique()}')
```

```
gender:[1 0]
ssc_b:[1 0]
hsc_b:[1 0]
hsc_s:[1 2 0]
degree_t:[2 0 1]
workex:[0 1]
specialisation:[1 0]
status:[1 0]
```

[30]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
for col in df.select_dtypes(include=['object']).columns:
    label_encoder = LabelEncoder()
    df[col] = label_encoder.fit_transform(dfcat[col])
    print(f'{col}:{df[col].unique()}')
```

```
gender:[1 0]
ssc_b:[1 0]
hsc_b:[1 0]
hsc_s:[1 2 0]
degree_t:[2 0 1]
workex:[0 1]
specialisation:[1 0]
status:[1 0]
```

[31]:
```
dfnum
```

[31]:

	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
0	67.00	91.00	58.00	55.0	58.80	270000.0
1	79.33	78.33	77.48	86.5	66.28	200000.0
2	65.00	68.00	64.00	75.0	57.80	250000.0
3	56.00	52.00	52.00	66.0	59.43	NaN
4	85.80	73.60	73.30	96.8	55.50	425000.0
..	…	…	…	…	…	…
210	80.60	82.00	77.60	91.0	74.49	400000.0

```
211  58.00  60.00     72.00     74.0  53.62  275000.0
212  67.00  67.00     73.00     59.0  69.72  295000.0
213  74.00  66.00     58.00     70.0  60.23  204000.0
214  62.00  58.00     53.00     89.0  60.22       NaN

[215 rows x 6 columns]
```

[32]:
```python
df
```

[32]:
```
     gender  ssc_p  ssc_b  hsc_p  hsc_b  hsc_s  degree_p  degree_t  workex  \
0         1  67.00      1  91.00      1      1     58.00         2       0
1         1  79.33      0  78.33      1      2     77.48         2       1
2         1  65.00      0  68.00      0      0     64.00         0       0
3         1  56.00      0  52.00      0      2     52.00         2       0
4         1  85.80      0  73.60      0      1     73.30         0       0
..      …      …    …      …    …      …        …       …
210       1  80.60      1  82.00      1      1     77.60         0       0
211       1  58.00      1  60.00      1      2     72.00         2       0
212       1  67.00      1  67.00      1      1     73.00         0       1
213       0  74.00      1  66.00      1      1     58.00         0       0
214       1  62.00      0  58.00      1      2     53.00         0       0

     etest_p  specialisation  mba_p  status    salary
0       55.0               1  58.80       1  270000.0
1       86.5               0  66.28       1  200000.0
2       75.0               0  57.80       1  250000.0
3       66.0               1  59.43       0  265000.0
4       96.8               0  55.50       1  425000.0
..       …               …    …      …         …
210     91.0               0  74.49       1  400000.0
211     74.0               0  53.62       1  275000.0
212     59.0               0  69.72       1  295000.0
213     70.0               1  60.23       1  204000.0
214     89.0               1  60.22       0  265000.0

[215 rows x 14 columns]
```

6 One - Hot Encoding

[33]:
```python
DF_encoded = pd.get_dummies(dfcat, columns=cat_col, drop_first=True)
for i in DF_encoded:
    print(f'{i}:{DF_encoded[i].unique()}')
```

```
gender_M:[1 0]
ssc_b_Others:[1 0]
hsc_b_Others:[1 0]
hsc_s_Commerce:[1 0]
```

```
hsc_s_Science:[0 1]
degree_t_Others:[0 1]
degree_t_Sci&Tech:[1 0]
workex_Yes:[0 1]
specialisation_Mkt&HR:[1 0]
status_Placed:[1 0]
```

[34]:
```
DF_encoded
```

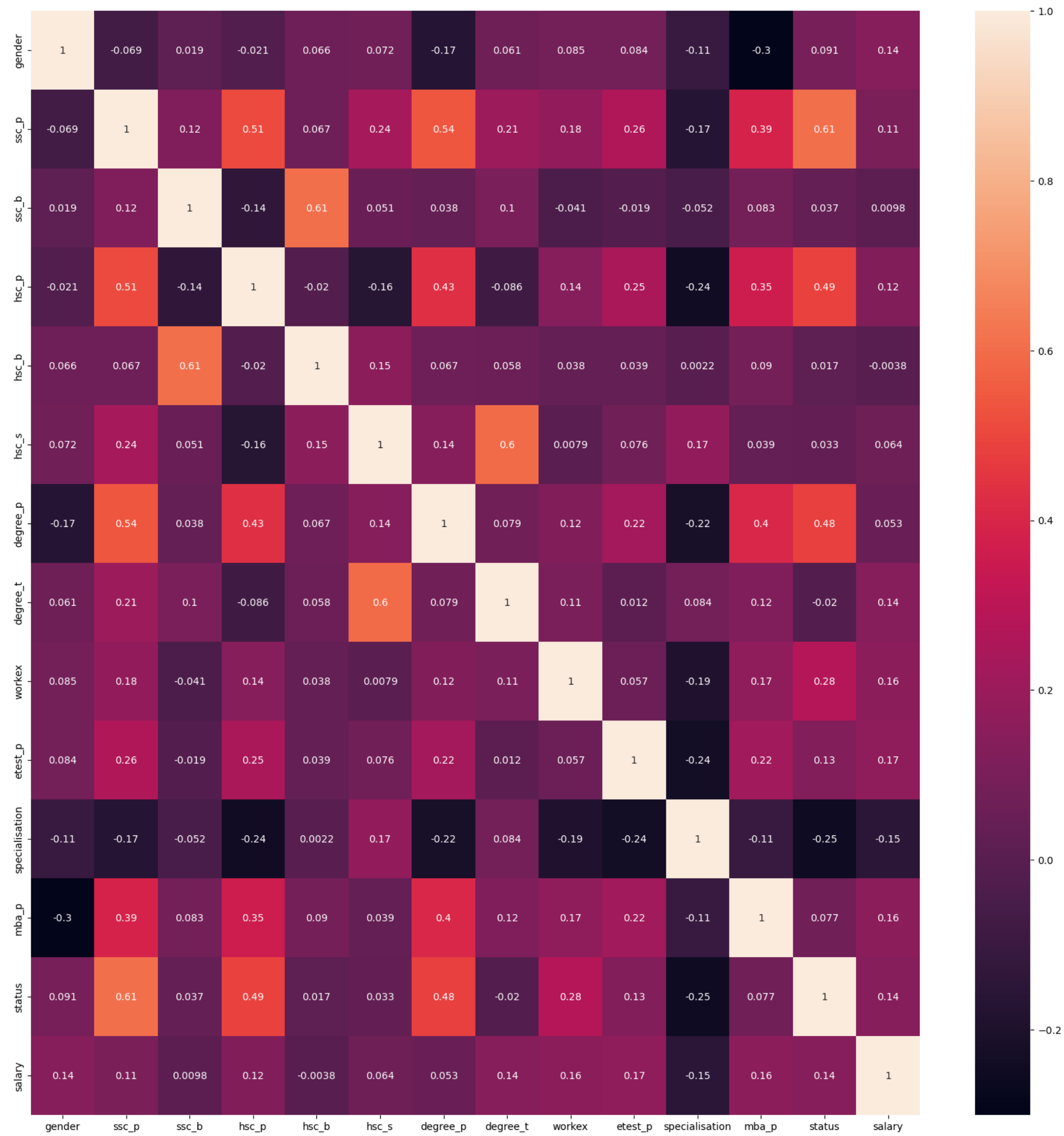
[34]:
```
     gender_M  ssc_b_Others  hsc_b_Others  hsc_s_Commerce  hsc_s_Science  \
0           1             1             1               1              0
1           1             0             1               0              1
2           1             0             0               0              0
3           1             0             0               0              1
4           1             0             0               1              0
..        …           …           …             …            …
210         1             1             1               1              0
211         1             1             1               0              1
212         1             1             1               1              0
213         0             1             1               1              0
214         1             0             1               0              1

     degree_t_Others  degree_t_Sci&Tech  workex_Yes  specialisation_Mkt&HR  \
0                  0                  1           0                      1
1                  0                  1           1                      0
2                  0                  0           0                      0
3                  0                  1           0                      1
4                  0                  0           0                      0
..               …                …         …                    …
210                0                  0           0                      0
211                0                  1           0                      0
212                0                  0           1                      0
213                0                  0           0                      1
214                0                  0           0                      1

     status_Placed
0                1
1                1
2                1
3                0
4                1
..             …
210              1
211              1
212              1
213              1
214              0
```

[215 rows x 10 columns]

7 Hear Map

[35]:
```python
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),fmt='.2g',annot=True)
```

[35]: <Axes: >



[36]:
```
df
```

[36]:
```
     gender  ssc_p  ssc_b  hsc_p  hsc_b  hsc_s  degree_p  degree_t  workex  \
0         1  67.00      1  91.00      1      1     58.00         2       0
1         1  79.33      0  78.33      1      2     77.48         2       1
2         1  65.00      0  68.00      0      0     64.00         0       0
3         1  56.00      0  52.00      0      2     52.00         2       0
4         1  85.80      0  73.60      0      1     73.30         0       0
..      …      …      …      …      …      …       …         …     …
210       1  80.60      1  82.00      1      1     77.60         0       0
211       1  58.00      1  60.00      1      2     72.00         2       0
212       1  67.00      1  67.00      1      1     73.00         0       1
213       0  74.00      1  66.00      1      1     58.00         0       0
214       1  62.00      0  58.00      1      2     53.00         0       0

     etest_p  specialisation  mba_p  status    salary
0       55.0               1  58.80       1  270000.0
1       86.5               0  66.28       1  200000.0
2       75.0               0  57.80       1  250000.0
3       66.0               1  59.43       0  265000.0
4       96.8               0  55.50       1  425000.0
..       …             …      …       …
210     91.0               0  74.49       1  400000.0
211     74.0               0  53.62       1  275000.0
212     59.0               0  69.72       1  295000.0
213     70.0               1  60.23       1  204000.0
214     89.0               1  60.22       0  265000.0

[215 rows x 14 columns]
```

8 Train Test Split

[40]:
```
from sklearn.model_selection import train_test_split
x=df.drop('status',axis=1)
y=df['status']
```

[41]:
```
xtrn,xtst,ytrn,ytst=train_test_split(x,y,train_size=0.2)
```

9 Remove Outlier drom data

[39]:
```
from scipy.stats import zscore
z_scores = zscore(df[num_col])
df_no_outliers = df[(z_scores < 3).all(axis=1)]
```

10 Decision Tree

[41]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
dt=DecisionTreeClassifier(class_weight='balanced')
param_grid={
    'max_depth':[2,5,7,11,13],
    'min_samples_split':[2,3,5,7],
    'min_samples_leaf':[2,3,4],
    #'max_feature':[0.2,0.3,0.5,0.7]
}

dt_grid=GridSearchCV(dt,param_grid,cv=4)
dt_grid.fit(xtrn,ytrn)
```

[41]:
```
GridSearchCV(cv=4, estimator=DecisionTreeClassifier(class_weight='balanced'),
             param_grid={'max_depth': [2, 5, 7, 11, 13],
                         'min_samples_leaf': [2, 3, 4],
                         'min_samples_split': [2, 3, 5, 7]})
```

[42]:
```
dt_grid.best_params_
```

[42]:
```
{'max_depth': 7, 'min_samples_leaf': 2, 'min_samples_split': 3}
```

[43]:
```
print("Accuracy Score: ", round(dt_grid.best_score_ * 100, 2), "%")
```

```
Accuracy Score:  78.86 %
```

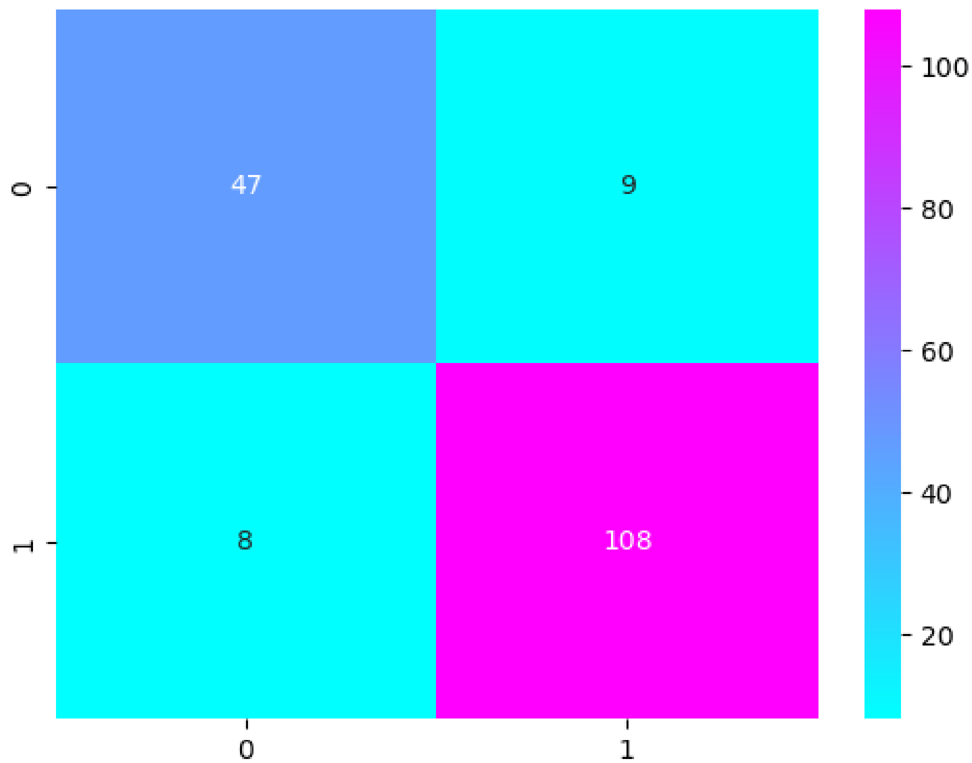
[44]:
```
ypred=dt_grid.predict(xtst)
```

11 Confusion Matrix

[45]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytst,ypred)
```

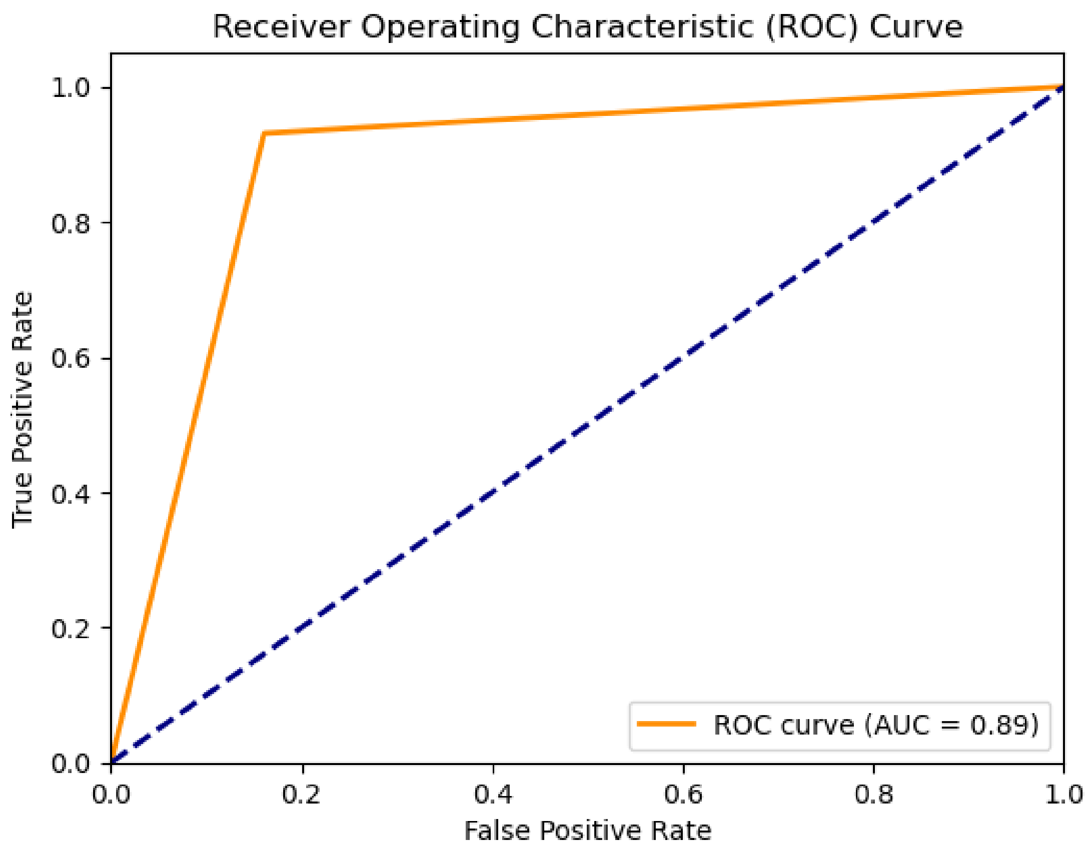
[46]:
```
sns.heatmap(data=cm,annot=True,fmt='d',cmap='cool')
```

[46]:
```
<Axes: >
```



12 ROC Curve

[47]:
```
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(ytst, ypred)

roc_auc = auc(fpr, tpr)


plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = %0.2f)' %␣
↪roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



13 Random Forest

[48]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Assuming you have your training data in 'xtrn' and labels in 'ytrn'

rf = RandomForestClassifier(class_weight='balanced')
param_grid = {
    'n_estimators': [10, 15, 17],
    'max_depth': [2, 5, 7, 11, 13, None],
    'min_samples_split': [2, 3, 5, 7],
    'min_samples_leaf': [2, 3, 4],
    'max_features': [0.2, 0.3, 0.5, 0.7]
}

rf_grid = GridSearchCV(rf, param_grid, cv=4)
rf_grid.fit(xtrn, ytrn)
```

[48]:
```
GridSearchCV(cv=4, estimator=RandomForestClassifier(class_weight='balanced'),
             param_grid={'max_depth': [2, 5, 7, 11, 13, None],
                         'max_features': [0.2, 0.3, 0.5, 0.7],
                         'min_samples_leaf': [2, 3, 4],
                         'min_samples_split': [2, 3, 5, 7],
                         'n_estimators': [10, 15, 17]})
```

[49]:
```
rf_grid.best_params_
```

[49]:
```
{'max_depth': 7,
 'max_features': 0.2,
 'min_samples_leaf': 2,
 'min_samples_split': 5,
 'n_estimators': 17}
```

[50]:
```
print("Accuracy Score: ", round(rf_grid.best_score_ * 100, 2), "%")
```

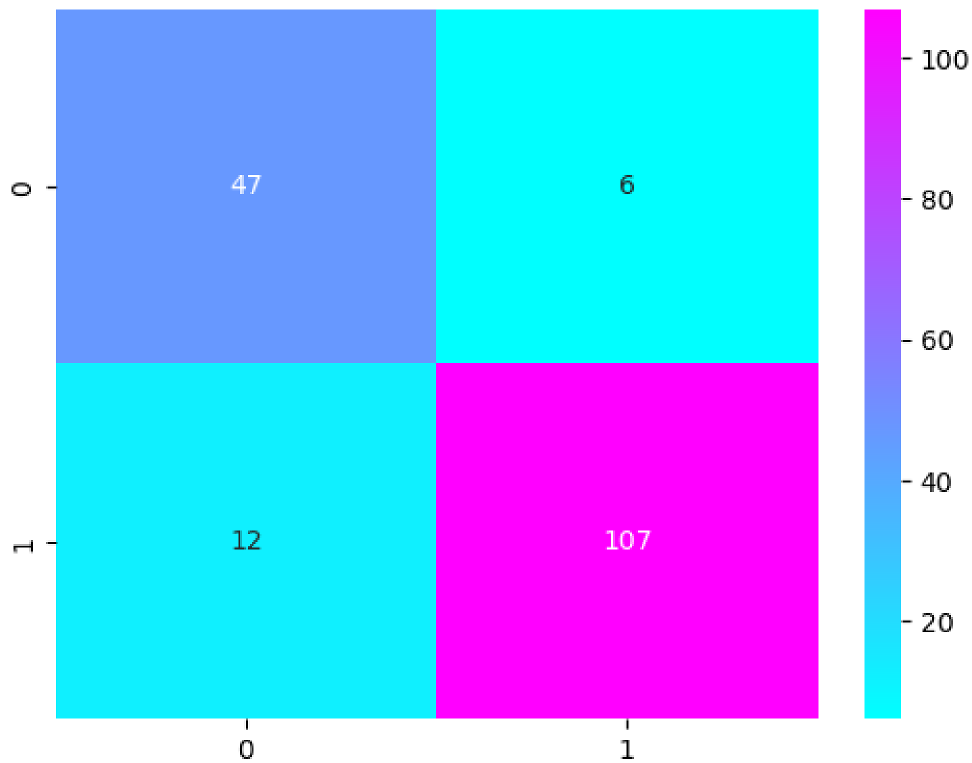
```
Accuracy Score:  95.45 %
```

[51]:
```
ypred=rf_grid.predict(xtst)
```

14 Confusion Matrix

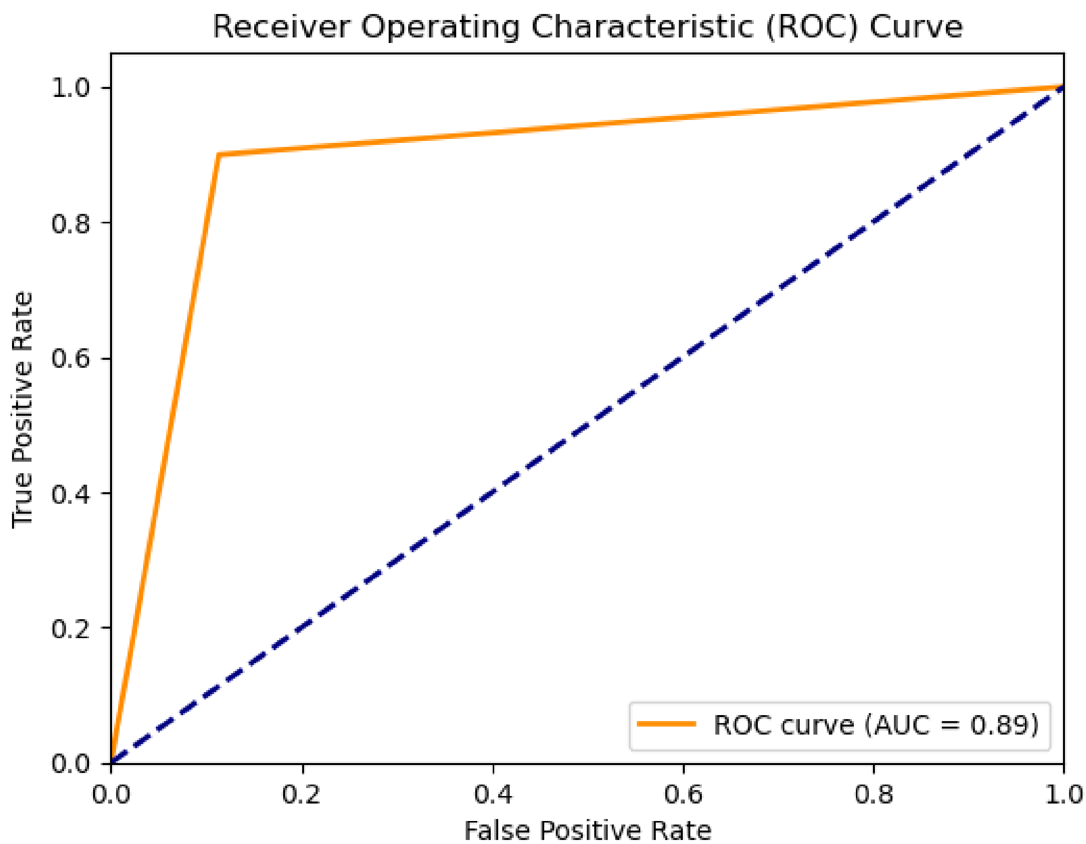
[57]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytst,ypred)
sns.heatmap(data=cm,annot=True,fmt='d',cmap='cool')
```

[57]: <Axes: >



15 ROC Curve

[58]:
```
fpr, tpr, thresholds = roc_curve(ytst, ypred)

roc_auc = auc(fpr, tpr)


plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = %0.2f)' %
 ↪roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



16 Logistic regression

[48]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
lg=LogisticRegression()

lg.fit(xtrn, ytrn)
ypred=lg.predict(xtst)
accuracy=round(accuracy_score(ytst, ypred),2)*100
print("Accuracy:", accuracy)
```

```
Accuracy: 84.0
```

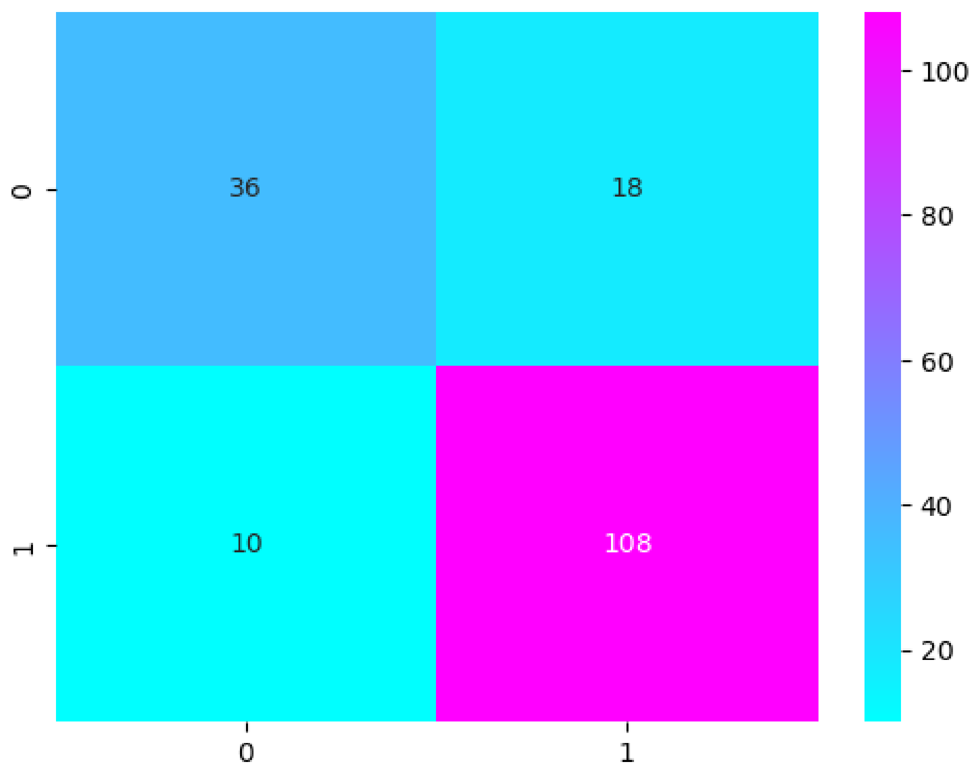
```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

17 Confusion Matrix

[49]:
```python
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytst,ypred)
sns.heatmap(data=cm,annot=True,fmt='d',cmap='cool')
```

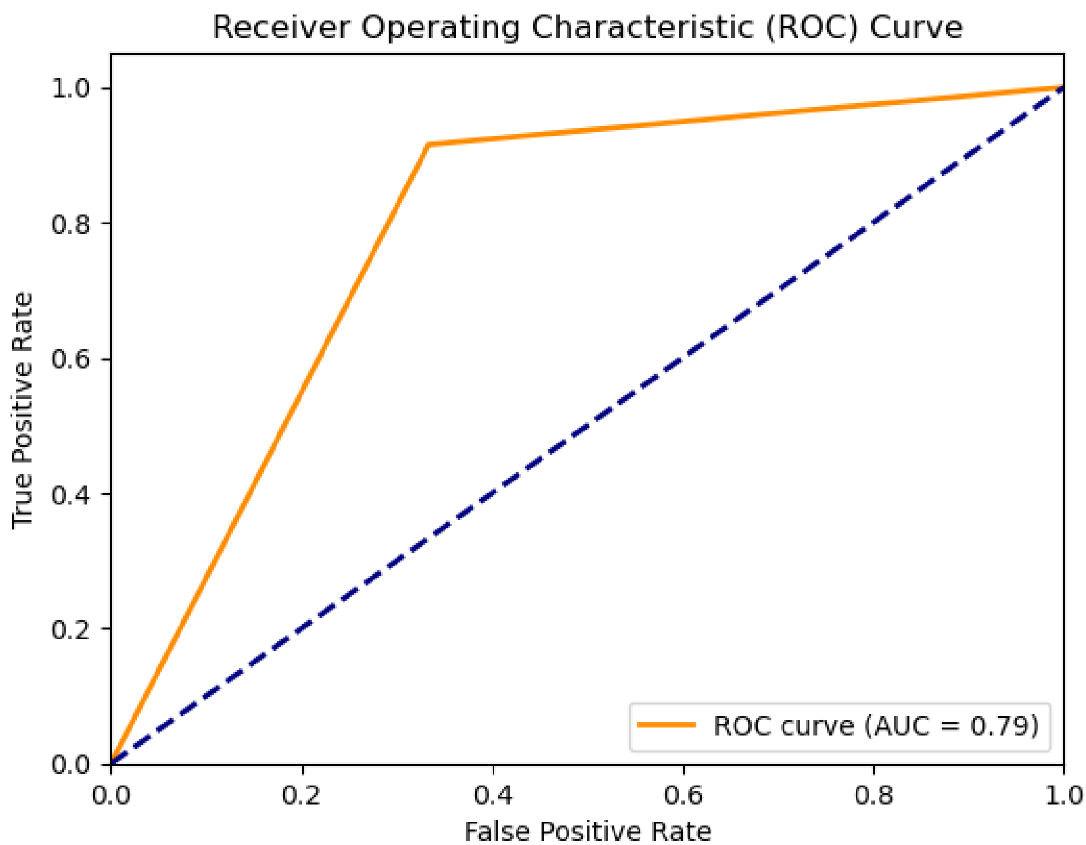
[49]:
```
<Axes: >
```



[ ]:
```python

```

[51]:
```python
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(ytst, ypred)

roc_auc = auc(fpr, tpr)


plt.figure()
```

```python
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = %0.2f)' %␣
↪roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



[ ]: