

TUGAS UAS SAINS DATA

NAMA : DWI AHMAD DZULHIJAH

EMAIL : 1818101@scholar.itn.ac.id (mailto:1818101@scholar.itn.ac.id)

Kampus asal : ITN Malang

Kampus ICEI : Universitas Diponegoro

Matkul : Sains Data (Data Science)

BUSINESS UNDESTANDING

MEMPERSIAPKAN LIBRARY

```
In [16]: 1 # data analysis and wrangling
2 import numpy as np
3 import pandas as pd
4 from scipy import stats
5 from scipy.stats import norm, skew
6 # visualization
7 import seaborn as sns
8 import matplotlib.pyplot as plt
9 import plotly.figure_factory as ff
10 import plotly.graph_objects as go
11 # machine Learning
12 from sklearn.preprocessing import LabelEncoder
13 from sklearn.preprocessing import MinMaxScaler
14 from sklearn.model_selection import train_test_split
15 from sklearn.linear_model import LogisticRegression
16 from sklearn.svm import SVC
17 from sklearn.ensemble import RandomForestClassifier
18 from sklearn.neighbors import KNeighborsClassifier
19 from sklearn.naive_bayes import GaussianNB
20 from sklearn.linear_model import Perceptron
21 from sklearn.linear_model import SGDClassifier
22 from sklearn.tree import DecisionTreeClassifier
23 from sklearn.metrics import accuracy_score, classification_report
24 from xgboost import XGBClassifier
25 from sklearn.feature_selection import SelectKBest
26 from sklearn.feature_selection import chi2
27 from sklearn.metrics import roc_curve
28 from sklearn.metrics import roc_auc_score
29 from sklearn.feature_selection import f_regression, mutual_info_regression
30 from xgboost import XGBRegressor
31 from xgboost import plot_importance
```

DATA (DATA WRANGLING)

MEMBACA DATA

```
In [4]: 1 Data = pd.read_csv("Placement_Data_Full_Class.csv")
```

```
In [99]: 1 Data.head(5)
```

```
Out[99]:
```

	sl_no	gender	ssc_p	hsc_p	degree_p	workex	etest_p	specialisation	mba_p	status
0	1	1	67.00	91.00	58.00	0	55.0	1	58.80	1
1	2	1	79.33	78.33	77.48	1	86.5	0	66.28	1
2	3	1	65.00	68.00	64.00	0	75.0	0	57.80	1
3	4	1	56.00	52.00	52.00	0	66.0	1	59.43	0
4	5	1	85.80	73.60	73.30	0	96.8	0	55.50	1

```
In [20]: 1 Data.shape
```

```
Out[20]: (215, 15)
```

Telaah TIPE DATA

In [6]:

```
1 Data.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
```

TELAHAH JENIS DATA

In [10]:

```
1 s = (Data.dtypes == 'object')
2 object_cols = list(s[s].index)
3 print("Categorical variables:")
4 print(object_cols)
```

Categorical variables:
['gender', 'ssc_b', 'hsc_b', 'hsc_s', 'degree_t', 'workex', 'specialisation', 'status']

In [11]:

```
1 x = (Data.dtypes == ('int64'))
2 integer_cols = list(x[x].index)
3 print("integer variables:")
4 print(integer_cols)
```

integer variables:
['sl_no']

In [12]:

```
1 x = (Data.dtypes == ('float64'))
2 float_cols = list(x[x].index)
3 print("float variables:")
4 print(float_cols)
```

float variables:
['ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p', 'salary']

In [13]:

```
1 print("numerical variables:")
2 numerical_cols=float_cols+integer_cols
3 print(numerical_cols)
```

numerical variables:
['ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p', 'salary', 'sl_no']

In [24]:

```
1 Data.columns
```

Out[24]: 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 [18]:

```
1 Data.describe()
```

Out[18]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

```
In [15]: 1 Data.isnull().sum()
```

```
Out[15]: 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
```

```
In [26]: 1 # getting the object columns
2 object_columns = Data.select_dtypes(include=['object']).columns
3
4 # iterating over each object type column
5 for col in object_columns:
6     print('-' * 40 + col + '-' * 40 , end='-')
7     display(Data[col].value_counts())
```

```
-----gender-----
```

```
M    139
F     76
Name: gender, dtype: int64
```

```
-----ssc_b-----
```

```
Central    116
Others     99
Name: ssc_b, dtype: int64
```

```
-----hsc_b-----
```

```
Others     131
Central    84
Name: hsc_b, dtype: int64
```

```
-----hsc_s-----
```

```
Commerce    113
Science     91
Arts        11
Name: hsc_s, dtype: int64
```

```
-----degree_t-----
```

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

```
-----workex-----
```

```
No      141
Yes      74
Name: workex, dtype: int64
```

```
-----specialisation-----
```

```
Mkt&Fin     120
Mkt&HR      95
Name: specialisation, dtype: int64
```

```
-----status-----
```

```
Placed      148
Not Placed   67
Name: status, dtype: int64
```

Data Preprocessing

Mengatasi Data Null

```
In [36]: 1 Data['salary'].fillna((Data['salary'].mean()), inplace=True)
```

```
In [37]: 1 Data.isnull().sum()
```

```
Out[37]: 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         0
dtype: int64
```

Label Encoding

```
In [41]: 1 label = LabelEncoder()
2 Data["gender"] = label.fit_transform(Data["gender"])
3 Data["ssc_b"] = label.fit_transform(Data["ssc_b"])
4 Data["hsc_b"] = label.fit_transform(Data["hsc_b"])
5 Data["hsc_s"] = label.fit_transform(Data["hsc_s"])
6 Data["degree_t"] = label.fit_transform(Data["degree_t"])
7 Data["workex"] = label.fit_transform(Data["workex"])
8 Data["specialisation"] = label.fit_transform(Data["specialisation"])
9 Data["status"] = label.fit_transform(Data["status"])
```

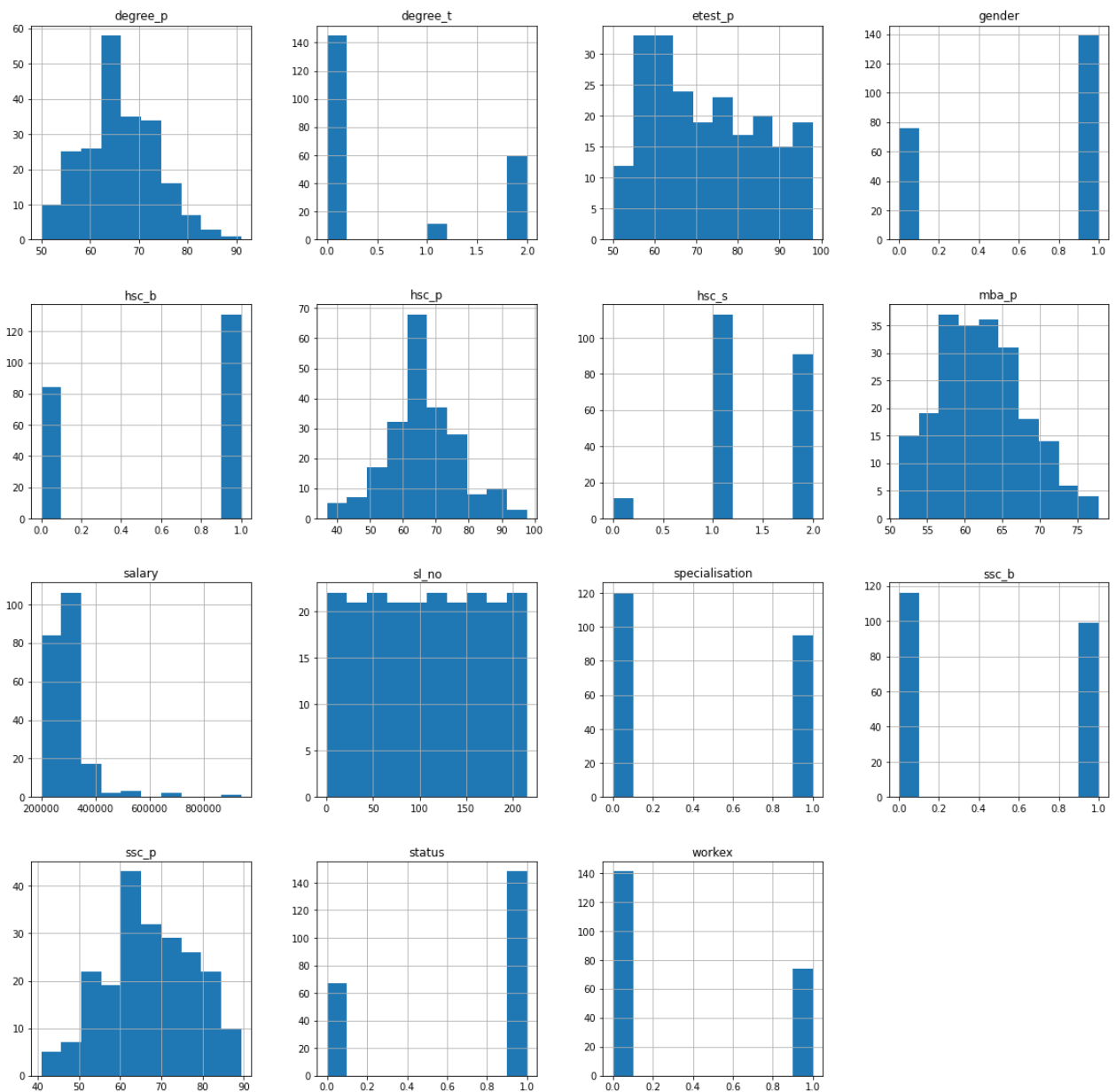
```
In [42]: 1 Data.head(10)
```

```
Out[42]:
```

	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	1	67.00	1	91.00	1	1	58.00	2	0	55.00	1	58.80	1	270000.000000
1	2	1	79.33	0	78.33	1	2	77.48	2	1	86.50	0	66.28	1	200000.000000
2	3	1	65.00	0	68.00	0	0	64.00	0	0	75.00	0	57.80	1	250000.000000
3	4	1	56.00	0	52.00	0	2	52.00	2	0	66.00	1	59.43	0	288655.405405
4	5	1	85.80	0	73.60	0	1	73.30	0	0	96.80	0	55.50	1	425000.000000
5	6	1	55.00	1	49.80	1	2	67.25	2	1	55.00	0	51.58	0	288655.405405
6	7	0	46.00	1	49.20	1	1	79.00	0	0	74.28	0	53.29	0	288655.405405
7	8	1	82.00	0	64.00	0	2	66.00	2	1	67.00	0	62.14	1	252000.000000
8	9	1	73.00	0	79.00	0	1	72.00	0	0	91.34	0	61.29	1	231000.000000
9	10	1	58.00	0	70.00	0	1	61.00	0	0	54.00	0	52.21	0	288655.405405

Eksplorasi Data (EDA)

```
In [43]: 1 Data.hist(figsize = (20, 20))  
2 plt.show()
```



```
In [96]: 1 cor=Data.corr()
2 plt.figure(figsize=(14,6))
3 sns.heatmap(cor,annot=True)
```

Out[96]: <matplotlib.axes._subplots.AxesSubplot at 0x1718283adc0>

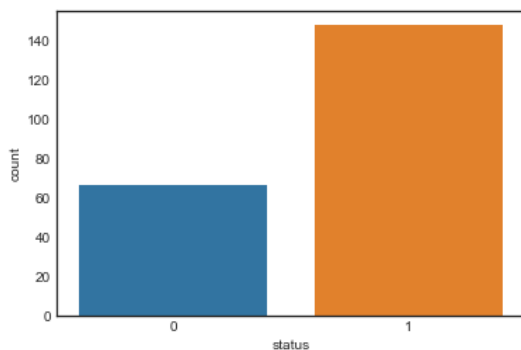


Analisis Pengalaman Kerja dan Status Penempatan

```
In [47]: 1 sns.countplot('status', data=Data)
```

C:\Users\dwiha\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
warnings.warn()

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x17182cce2e0>



```
In [49]: 1 Data['gender'].value_counts()
```

```
Out[49]: 1    139
0     76
Name: gender, dtype: int64
```

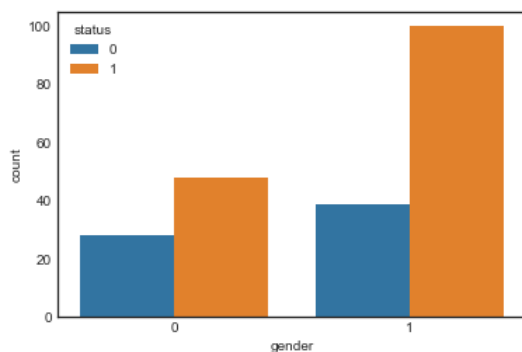
```
In [52]: 1 df = pd.DataFrame(Data.groupby(['gender','status'])['status'].count())
2 df
```

```
Out[52]:
```

		status	
gender	status		
0	0	28	
	1	48	
1	0	39	
	1	100	

```
In [54]: 1 sns.countplot(x='gender', hue='status', data=Data)
```

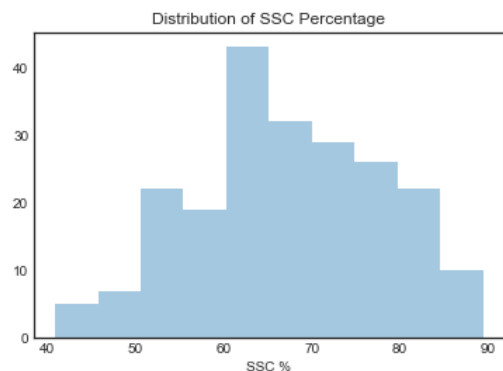
```
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x171828e26d0>
```



```
In [56]: 1 sns.distplot(Data['ssc_p'], kde=False)
2 plt.title('Distribution of SSC Percentage')
3 plt.xlabel('SSC %')
```

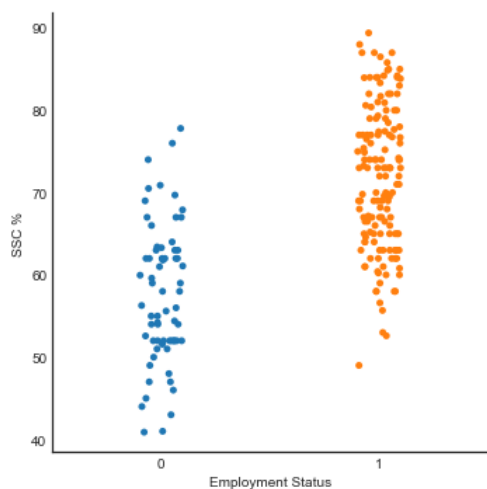
C:\Users\dwiah\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

```
Out[56]: Text(0.5, 0, 'SSC %')
```



```
In [58]: 1 sns.catplot(y='ssc_p', x='status', data=Data)
2 plt.xlabel('Employment Status')
3 plt.ylabel('SSC %')
```

```
Out[58]: Text(9.549999999999997, 0.5, 'SSC %')
```



```
In [59]: 1 Data['ssc_b'].value_counts()
```

```
Out[59]: 0    116
         1     99
         Name: ssc_b, dtype: int64
```

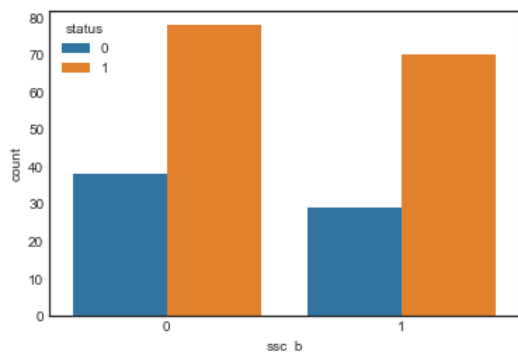
```
In [60]: 1 df = pd.DataFrame(Data.groupby(['ssc_b', 'status'])['status'].count())
         2 df
```

```
Out[60]:
```

		status	
ssc_b	status		
0	0	38	
	1	78	
1	0	29	
	1	70	

```
In [62]: 1 sns.countplot(x='ssc_b', hue='status', data=Data)
```

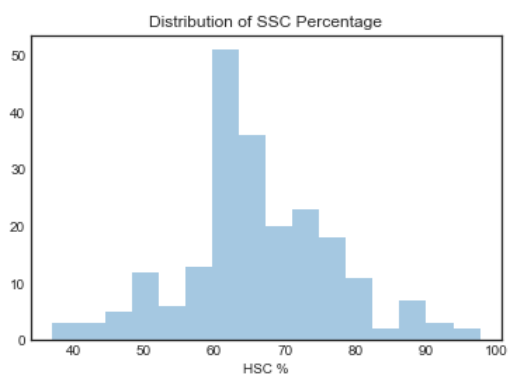
```
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x171808a1250>
```



```
In [63]: 1 sns.distplot(Data['hsc_p'], kde=False)
         2 plt.title('Distribution of SSC Percentage')
         3 plt.xlabel('HSC %')
```

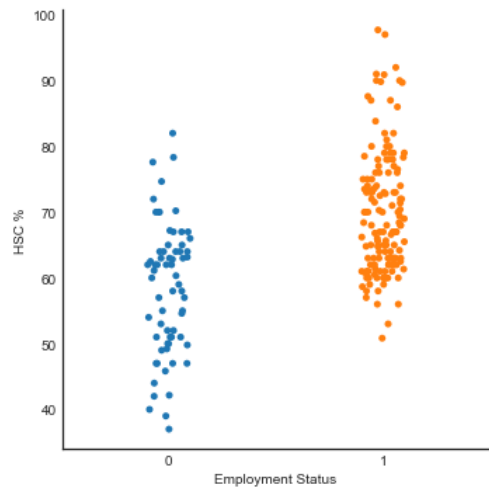
C:\Users\dwiah\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

```
Out[63]: Text(0.5, 0, 'HSC %')
```




```
In [65]: 1 sns.catplot(y='hsc_p', x='status', data=Data)
2         plt.xlabel('Employment Status')
3         plt.ylabel('HSC %')
```

Out[65]: Text(3.924999999999997, 0.5, 'HSC %')



```
In [66]: 1 Data['hsc_b'].value_counts()
```

Out[66]: 1 131
0 84
Name: hsc_b, dtype: int64

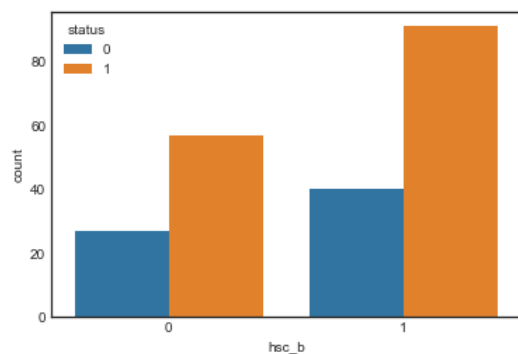
```
In [67]: 1 df = pd.DataFrame(Data.groupby(['hsc_b', 'status'])['status'].count())
2         df
```

Out[67]:

status		
hsc_b	status	
0	0	27
	1	57
1	0	40
	1	91

```
In [68]: 1 sns.countplot(x='hsc_b', hue='status', data=Data)
```

Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0x171822753d0>



```
In [69]: 1 Data['hsc_s'].value_counts()
```

```
Out[69]: 1    113
         2     91
         0     11
         Name: hsc_s, dtype: int64
```

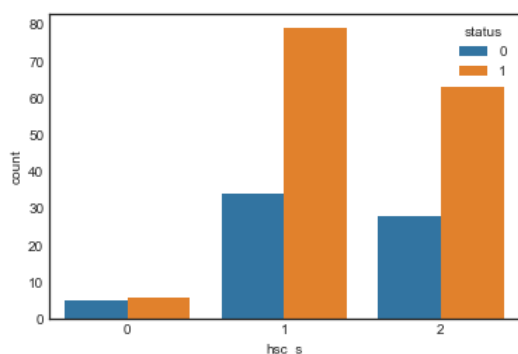
```
In [70]: 1 df = pd.DataFrame(Data.groupby(['hsc_s', 'status'])['status'].count())
         2 df
```

```
Out[70]:
```

		status	
hsc_s	status		
0	0	5	
	1	6	
1	0	34	
	1	79	
2	0	28	
	1	63	

```
In [71]: 1 sns.countplot(x='hsc_s', hue='status', data=Data)
```

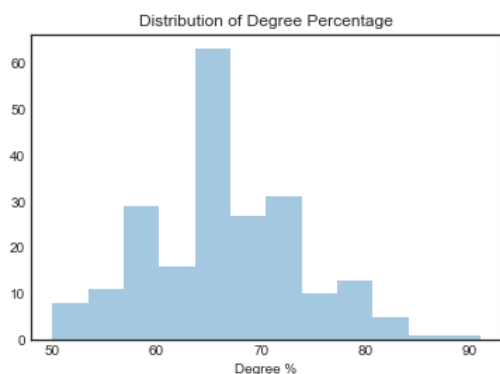
```
Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x17180876e50>
```



```
In [72]: 1 sns.distplot(Data['degree_p'], kde=False)
         2 plt.title('Distribution of Degree Percentage')
         3 plt.xlabel('Degree %')
```

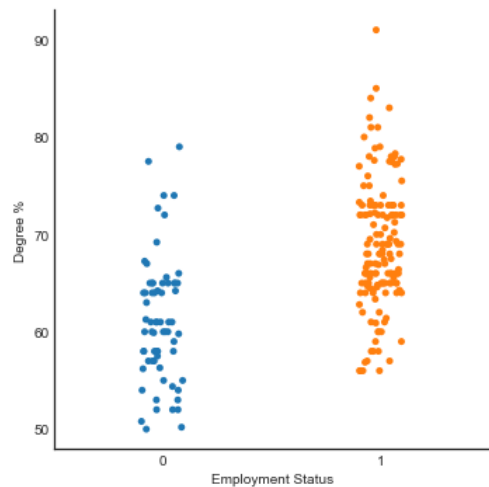
C:\Users\dwiah\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

```
Out[72]: Text(0.5, 0, 'Degree %')
```



```
In [73]: 1 sns.catplot(y='degree_p', x='status', data=Data)
2 plt.xlabel('Employment Status')
3 plt.ylabel('Degree %')
```

Out[73]: Text(9.54999999999997, 0.5, 'Degree %')



```
In [74]: 1 Data['degree_t'].value_counts()
```

Out[74]: 0 145
2 59
1 11
Name: degree_t, dtype: int64

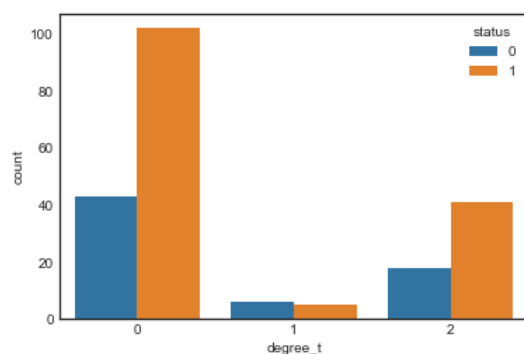
```
In [75]: 1 df = pd.DataFrame(Data.groupby(['degree_t', 'status'])['status'].count())
2 df
```

Out[75]:

status		
degree_t	status	
0	0	43
	1	102
1	0	6
	1	5
2	0	18
	1	41

```
In [76]: 1 sns.countplot(x='degree_t', hue='status', data=Data)
```

Out[76]: <matplotlib.axes._subplots.AxesSubplot at 0x1718089cc70>



```
In [77]: 1 Data['workex'].value_counts()
```

```
Out[77]: 0    141
         1     74
         Name: workex, dtype: int64
```

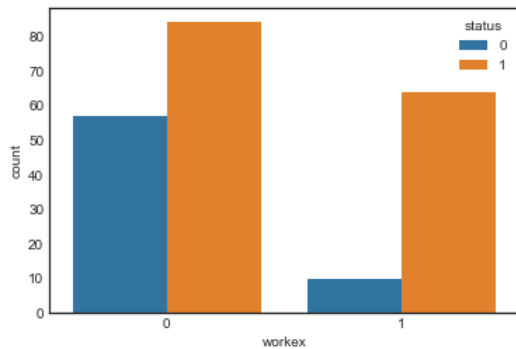
```
In [78]: 1 df = pd.DataFrame(Data.groupby(['workex', 'status'])['status'].count())
         2 df
```

```
Out[78]:
```

		status	
workex	status		
0	0	57	
	1	84	
1	0	10	
	1	64	

```
In [79]: 1 sns.countplot(x='workex', hue='status', data=Data)
```

```
Out[79]: <matplotlib.axes._subplots.AxesSubplot at 0x17180765b80>
```



```
In [80]: 1 sns.distplot(Data['etest_p'], kde=False)
         2 plt.title('Distribution of MBA Percentage')
         3 plt.xlabel('Employment Test %')
```

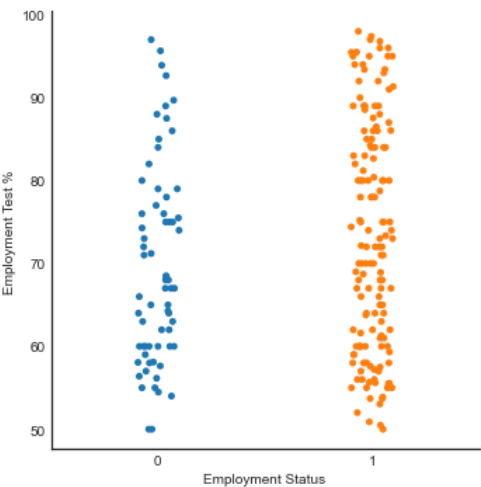
C:\Users\dwiah\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

```
Out[80]: Text(0.5, 0, 'Employment Test %')
```



```
In [81]: 1 sns.catplot(y='etest_p', x='status', data=Data)
2 plt.xlabel('Employment Status')
3 plt.ylabel('Employment Test %')
```

Out[81]: Text(3.924999999999997, 0.5, 'Employment Test %')



```
In [82]: 1 Data['specialisation'].value_counts()
```

Out[82]: 0 120
1 95
Name: specialisation, dtype: int64

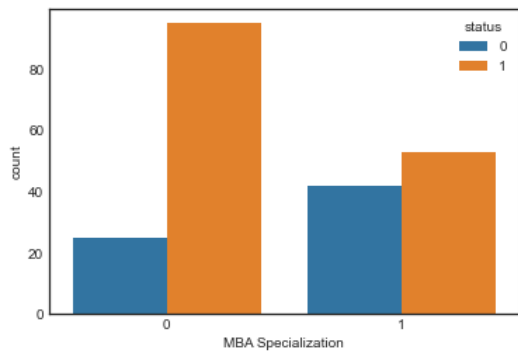
```
In [83]: 1 df = pd.DataFrame(Data.groupby(['specialisation', 'status'])['status'].count())
2 df
```

Out[83]:

		status	
specialisation	status		
0	0	25	
	1	95	
1	0	42	
	1	53	

```
In [84]: 1 sns.countplot(x='specialisation', hue='status', data=Data)
        2 plt.xlabel('MBA Specialization')
```

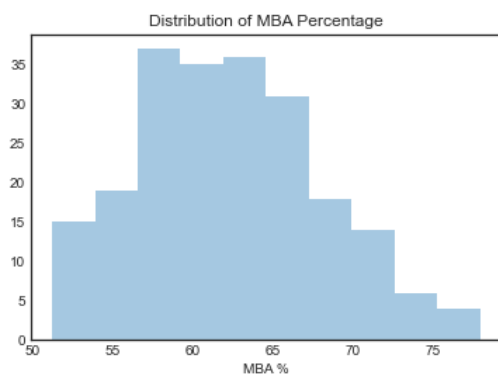
Out[84]: Text(0.5, 0, 'MBA Specialization')



```
In [85]: 1 sns.distplot(Data['mba_p'], kde=False)
        2 plt.title('Distribution of MBA Percentage')
        3 plt.xlabel('MBA %')
```

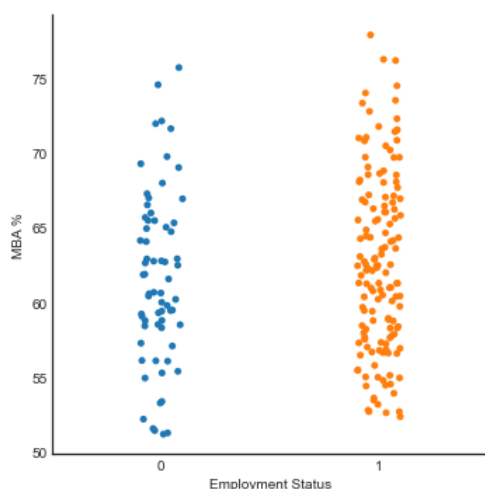
C:\Users\dwiah\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

Out[85]: Text(0.5, 0, 'MBA %')



```
In [86]: 1 sns.catplot(y='mba_p', x='status', data=Data)
        2 plt.xlabel('Employment Status')
        3 plt.ylabel('MBA %')
```

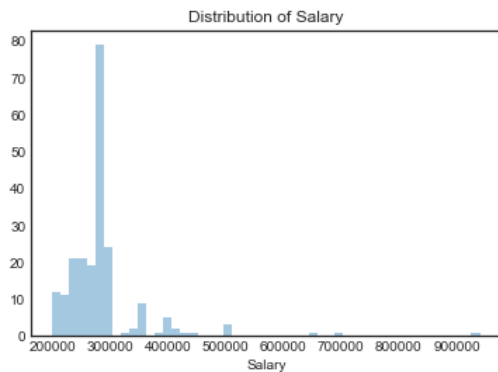
Out[86]: Text(9.549999999999997, 0.5, 'MBA %')



```
In [87]: 1 sns.distplot(Data['salary'], kde=False)
2         plt.title('Distribution of Salary')
3         plt.xlabel('Salary')
```

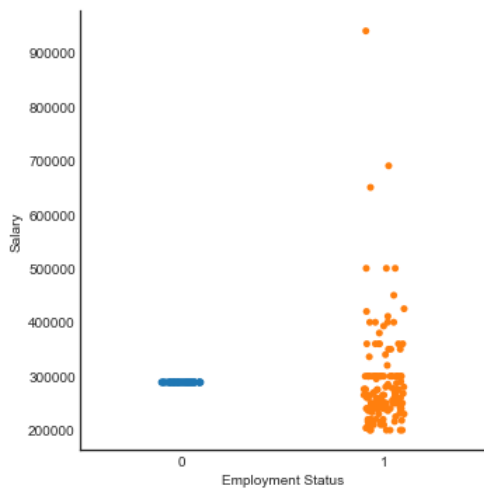
C:\Users\dwiah\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

Out[87]: Text(0.5, 0, 'Salary')



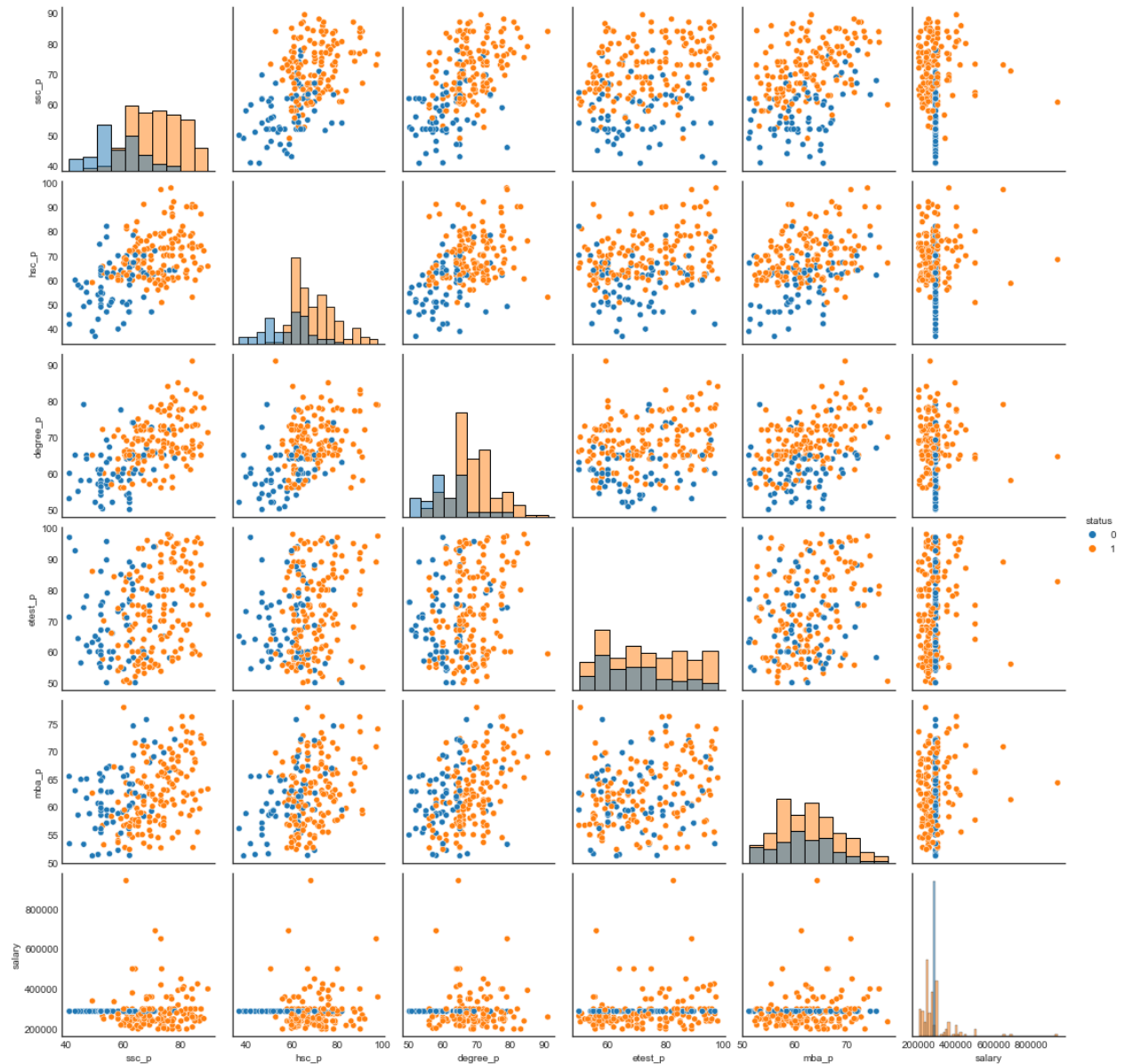
```
In [88]: 1 sns.catplot(y='salary', x='status', data=Data)
2         plt.xlabel('Employment Status')
3         plt.ylabel('Salary')
```

Out[88]: Text(-12.95000000000003, 0.5, 'Salary')



```
In [89]: 1 sns.pairplot(data=Data[['ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p', 'salary', 'status']], hue="status", diag_ki
```

```
Out[89]: <seaborn.axisgrid.PairGrid at 0x17181a5c940>
```

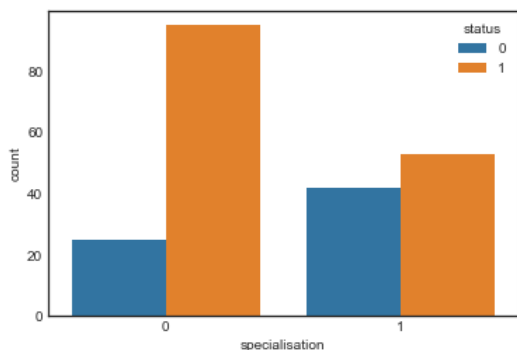



```
In [92]: 1 sns.countplot("specialisation", hue="status", data=Data)
```

C:\Users\dwiah\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

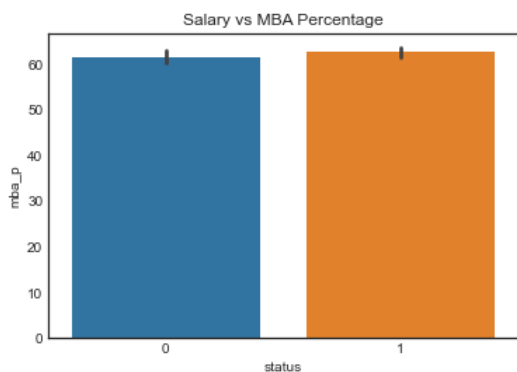
warnings.warn(

```
Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x171828371f0>
```



```
In [95]: 1 sns.barplot(x="status", y="mba_p", data=Data)
2 plt.title("Salary vs MBA Percentage")
```

```
Out[95]: Text(0.5, 1.0, 'Salary vs MBA Percentage')
```



```
In [101]: 1 Data.dtypes
```

```
Out[101]: sl_no          int64
gender          int32
ssc_p          float64
hsc_p          float64
degree_p       float64
workex         int32
etest_p       float64
specialisation  int32
mba_p          float64
status         int32
dtype: object
```

```
In [102]: 1 data_clf = Data.copy()
2 data_reg = Data.copy()
```

Data Modelling

DECISION TREE CLASSIFIER

```
In [103]: 1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score, classification_report
```

```
In [105]: 1 # Separating Features and Target
2 X = data_clf[['gender', 'ssc_p', 'hsc_p', 'degree_p', 'workex', 'etest_p', 'specialisation', 'mba_p']]
3 y = data_clf['status']
```

```
In [106]: 1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
In [107]: 1 dtree = DecisionTreeClassifier(criterion='entropy')
2 dtree.fit(X_train, y_train)
3 y_pred = dtree.predict(X_test)
```

```
In [108]: 1 accuracy_score(y_test, y_pred)
```

Out[108]: 0.8307692307692308

```
In [109]: 1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.88	0.62	0.73	24
1	0.81	0.95	0.88	41
accuracy			0.83	65
macro avg	0.85	0.79	0.80	65
weighted avg	0.84	0.83	0.82	65

RANDOM FOREST ALGORITHM

```
In [114]: 1 #Using Random Forest Algorithm
2 random_forest = RandomForestClassifier(n_estimators=100)
3 random_forest.fit(X_train, y_train)
4 y_pred = random_forest.predict(X_test)
```

```
In [115]: 1 accuracy_score(y_test, y_pred)
```

Out[115]: 0.8615384615384616

```
In [116]: 1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.62	0.77	24
1	0.82	1.00	0.90	41
accuracy			0.86	65
macro avg	0.91	0.81	0.84	65
weighted avg	0.89	0.86	0.85	65

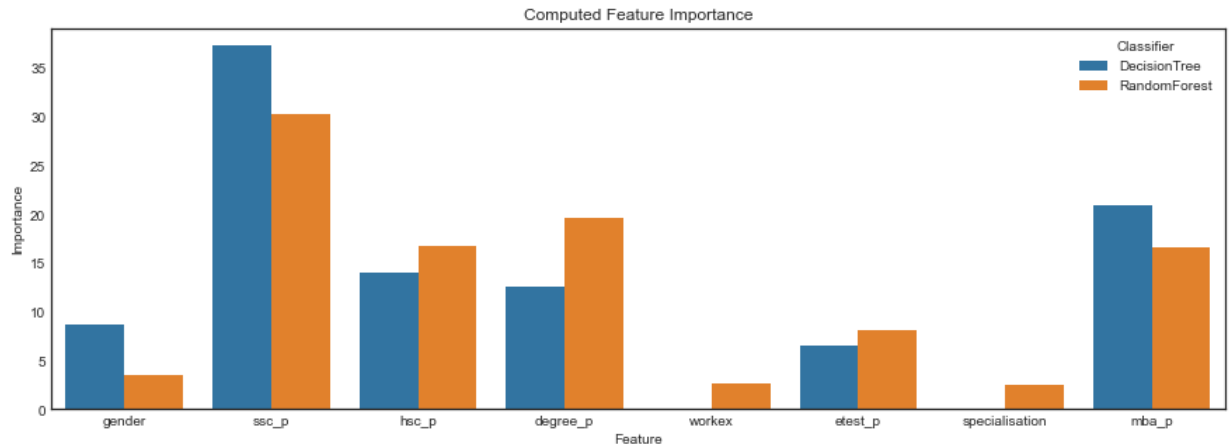
Feature Selection untuk Regresi Linear

```
In [117]: 1 rows = list(X.columns)
2 imp = pd.DataFrame(np.zeros(6*len(rows)).reshape(2*len(rows), 3))
3 imp.columns = ["Classifier", "Feature", "Importance"]
4 #Add Rows
5 for index in range(0, 2*len(rows), 2):
6     imp.iloc[index] = ["DecisionTree", rows[index//2], (100*dtree.feature_importances_[index//2])]
7     imp.iloc[index + 1] = ["RandomForest", rows[index//2], (100*random_forest.feature_importances_[index//2])]
```

```
In [118]: 1 plt.figure(figsize=(15,5))
2 sns.barplot("Feature", "Importance", hue="Classifier", data=imp)
3 plt.title("Computed Feature Importance")
4 plt.show()
```

C:\Users\dwiwh\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other argument s without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



```
In [125]: 1 # Seperating Features and Target
2 X = data_clf[['gender', 'ssc_p', 'hsc_p', 'degree_p', 'workex', 'etest_p', 'specialisation', 'mba_p']]
3 y = data_clf['status']
4 #Reverse Mapping and making Categorical
5 X["gender"] = pd.Categorical(X.gender.map({0:"M",1:"F"}))
6 X["workex"] = pd.Categorical(X.workex.map({0:"No",1:"Yes"}))
7 X["specialisation"] = pd.Categorical(X.specialisation.map({0:"Mkt&HR",1:"Mkt&Fin"}))
```

```
In [134]: 1 X.head(10)
```

Out[134]:

	ssc_p	hsc_p	degree_p	etest_p	mba_p	gender_F	gender_M	workex_No	workex_Yes	specialisation_Mkt&Fin	specialisation_Mkt&HR
0	67.00	91.00	58.00	55.00	58.80	1	0	1	0	1	0
1	79.33	78.33	77.48	86.50	66.28	1	0	0	1	0	1
2	65.00	68.00	64.00	75.00	57.80	1	0	1	0	0	1
3	56.00	52.00	52.00	66.00	59.43	1	0	1	0	1	0
4	85.80	73.60	73.30	96.80	55.50	1	0	1	0	0	1
5	55.00	49.80	67.25	55.00	51.58	1	0	0	1	0	1
6	46.00	49.20	79.00	74.28	53.29	0	1	1	0	0	1
7	82.00	64.00	66.00	67.00	62.14	1	0	0	1	0	1
8	73.00	79.00	72.00	91.34	61.29	1	0	1	0	0	1
9	58.00	70.00	61.00	54.00	52.21	1	0	1	0	0	1

One-Hot Encoding

```
In [126]: 1 X = pd.get_dummies(X)
2 colmunnn_names = X.columns.to_list()
```

```
In [127]: 1 from sklearn.preprocessing import MinMaxScaler
2 scaler = MinMaxScaler()
3 X_scaled = scaler.fit_transform(X)
```

```
In [128]: 1 #Train Test Split
2 X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3)
```

Logistic Regression

```
In [129]: 1 from sklearn.linear_model import LogisticRegression
```

```
In [130]: 1 logistic_reg = LogisticRegression()  
2 logistic_reg.fit(X_train, y_train)  
3 y_pred = logistic_reg.predict(X_test)
```

```
In [131]: 1 accuracy_score(y_test, y_pred)
```

Out[131]: 0.8615384615384616

```
In [133]: 1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.53	0.69	19
1	0.84	1.00	0.91	46
accuracy			0.86	65
macro avg	0.92	0.76	0.80	65
weighted avg	0.88	0.86	0.85	65