A Project Report

submitted as part of the course

Information Visualization (CSE3044)

School of Computer Science and Engineering

VIT Chennai

Winter Semester 2020-2021

Course Faculty: Dr. Arun Kumar Sivaraman

PROJECT:

CAMPUS RECRUITMENT

NAME: SANJANA ALAHAM

REG NO: 19MIA1055

ABSTRACT

Campus recruitment is a strategy for sourcing, engaging and hiring young talent for internship and entry-level positions. College recruiting is typically a tactic for medium- to large-sized companies with high-volume recruiting needs, but can range from small efforts (like working with university career centers to source potential candidates) to large-scale operations (like visiting a wide array of colleges and attending recruiting events throughout the spring and fall semester). Campus recruitment often involves working with university career services centers and attending career fairs to meet in-person with college students and recent graduates.

INTRODUCTION

This data set consists of Placement data of students . It includes secondary and higher secondary school percentage and specialization. It also includes degree specialization, type and Work experience and salary offers to the placed students.

Questions:

1. Which factor influenced a candidate in getting placed?
2. Does percentage matter for one to get placed?
3. Which degree specialization is much demanded by corporations?
4. Does work experience matter for one to get placed?
5. Play with the data conducting all statistical tests.

In this project we are going to visualize the chances and possibility for a candidate getting placed in a company based on the factors given below:

1. Work experience

2. 10th grade percentage

3. 12th Grade percentage

4. Degree percentage

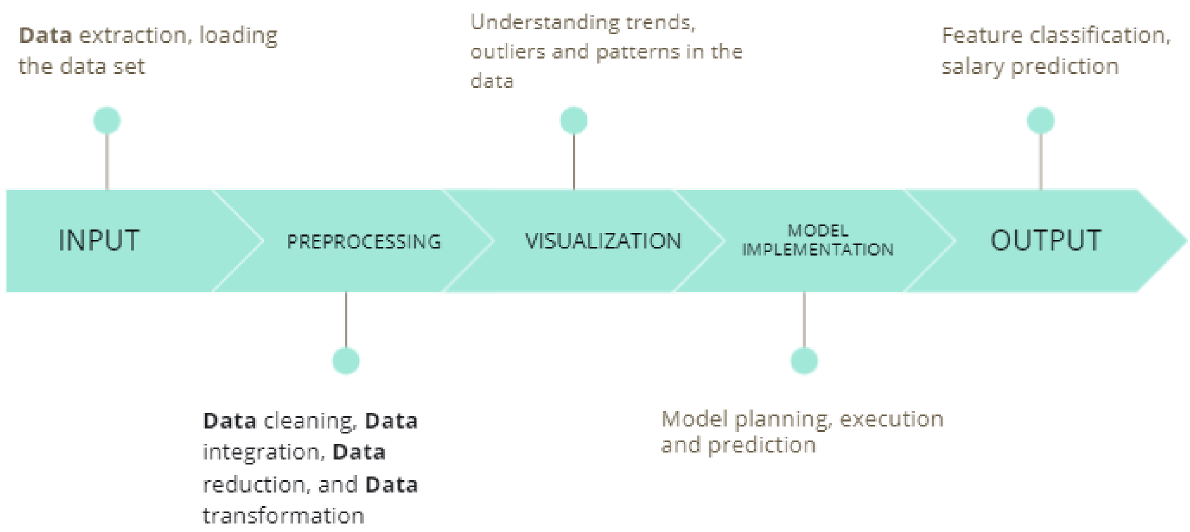
5. Gender

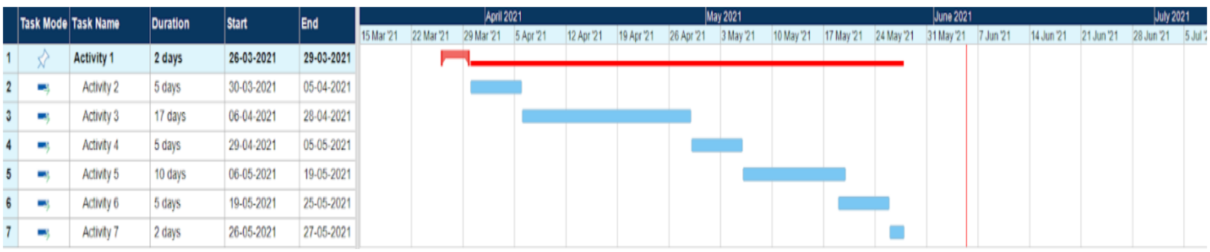
6. Field of Degree

PROJECT MODULES

MODULE-1

It is the work flow i.e., the process of our project





	Task Mode	Task Name	Duration	Start	End
1		**Activity 1**	**2 days**	**26-03-2021**	**29-03-2021**
2		Activity 2	5 days	30-03-2021	05-04-2021
3		Activity 3	17 days	06-04-2021	28-04-2021
4		Activity 4	5 days	29-04-2021	05-05-2021
5		Activity 5	10 days	06-05-2021	19-05-2021
6		Activity 6	5 days	19-05-2021	25-05-2021
7		Activity 7	2 days	26-05-2021	27-05-2021

MODULE-2

Overview of the data set

Overview Reproduction Warnings 1

Dataset statistics

Number of variables	14
Number of observations	215
Missing cells	67
Missing cells (%)	2.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	117.3 KiB
Average record size in memory	558.6 B

Variable types

CAT	7
NUM	6
BOOL	1

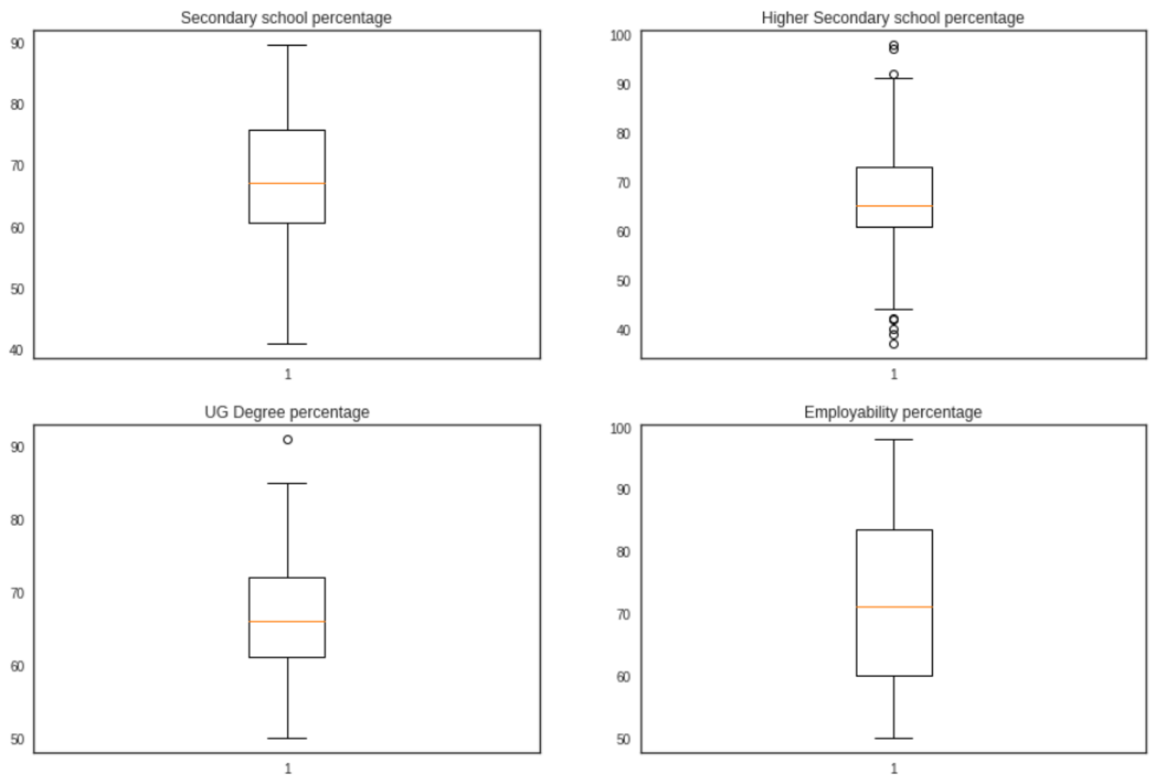
MODULE-3

- 67 Missing values in Salary for students who didn't get placed. **NaN Value needs to be filled**.
- **Data is not scaled**. Salary column ranges from 200k-940k, rest of numerical columns are percentages.
- 300k at 75th Percentile goes all the way up to 940k max, in Salary (high skewnwss). Thus, **outliers at high salary end**.

Outlier Detection

In [91]:
```
plt.figure(figsize = (15, 10))
plt.style.use('seaborn-white')
ax=plt.subplot(221)
plt.boxplot(data['ssc_p'])
ax.set_title('Secondary school percentage')
ax=plt.subplot(222)
plt.boxplot(data['hsc_p'])
ax.set_title('Higher Secondary school percentage')
ax=plt.subplot(223)
plt.boxplot(data['degree_p'])
ax.set_title('UG Degree percentage')
ax=plt.subplot(224)
plt.boxplot(data['etest_p'])
ax.set_title('Employability percentage')
```

Out[91]:
```
Text(0.5, 1.0, 'Employability percentage')
```

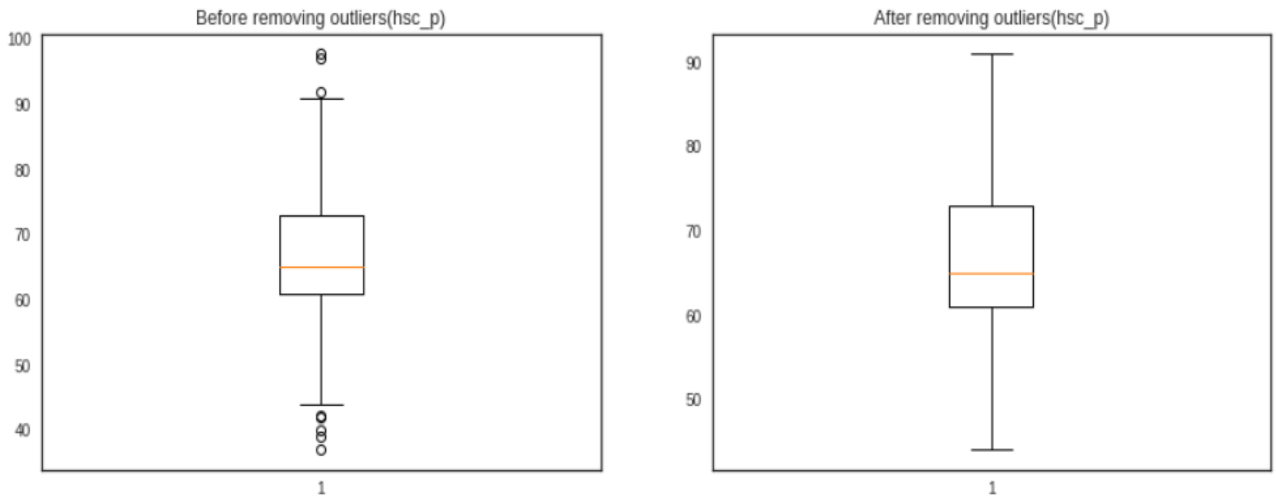


In [92]:
```
Q1 = data['hsc_p'].quantile(0.25)
Q3 = data['hsc_p'].quantile(0.75)
IQR = Q3 - Q1    #IQR is interquartile range.

filter = (data['hsc_p'] >= Q1 - 1.5 * IQR) & (data['hsc_p'] <= Q3 + 1.5 *IQR)
data_new=data.loc[filter]
```

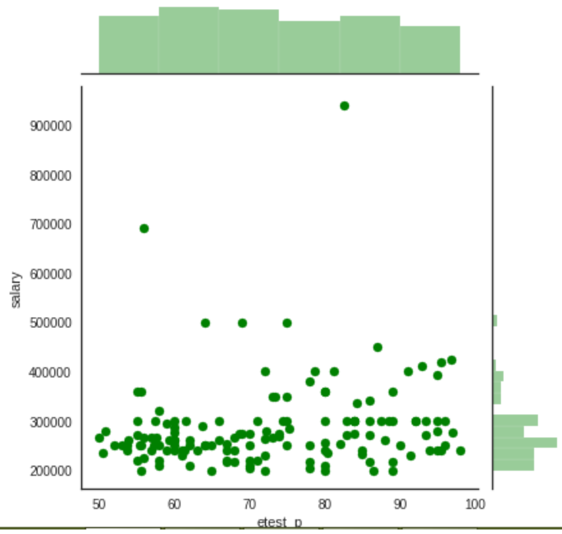
In [93]:
```
plt.figure(figsize = (15, 5))
plt.style.use('seaborn-white')
ax=plt.subplot(121)
plt.boxplot(data['hsc_p'])
ax.set_title('Before removing outliers(hsc_p)')
ax=plt.subplot(122)
plt.boxplot(data_new['hsc_p'])
ax.set_title('After removing outliers(hsc_p)')
```

Out[93]:
```
Text(0.5, 1.0, 'After removing outliers(hsc_p)')
```



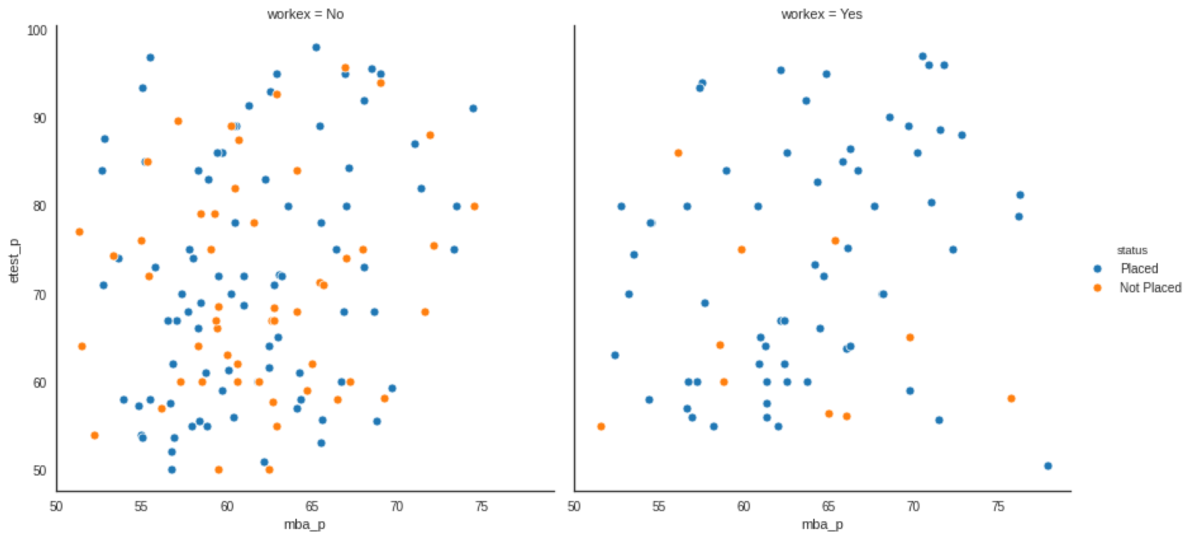

```
In [94]: sns.jointplot(x='etest_p',y='salary',data=data_new,color='green')
```

```
Out[94]: <seaborn.axisgrid.JointGrid at 0x7fbdb77a5710>
```




```
In [95]: g=sns.FacetGrid(data=data_new,col='workex',hue='status',height=6)
         g.map(sns.scatterplot,'mba_p','etest_p',label='status')
         g.add_legend()
```

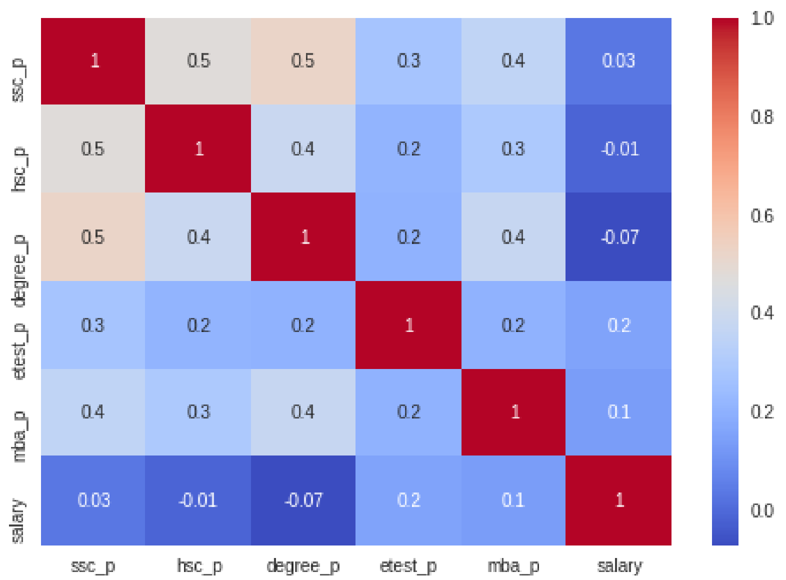
```
Out[95]: <seaborn.axisgrid.FacetGrid at 0x7fbdb74bee80>
```



Coorelation between academic percentages

In [96]:
```
p=data_new.corr()
sns.heatmap(p,cmap='coolwarm',annot=True,fmt='.1g')
```

Out[96]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdb7410eb8>



Vizualizing individual features

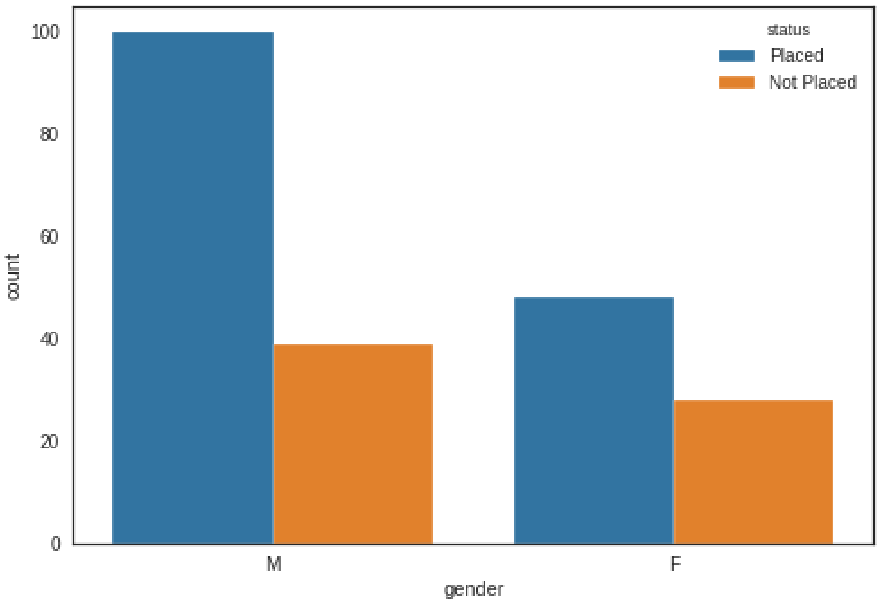
Feature: Gender

Does gender affect placements?

?]:
```
data.gender.value_counts()
# Almost double
```

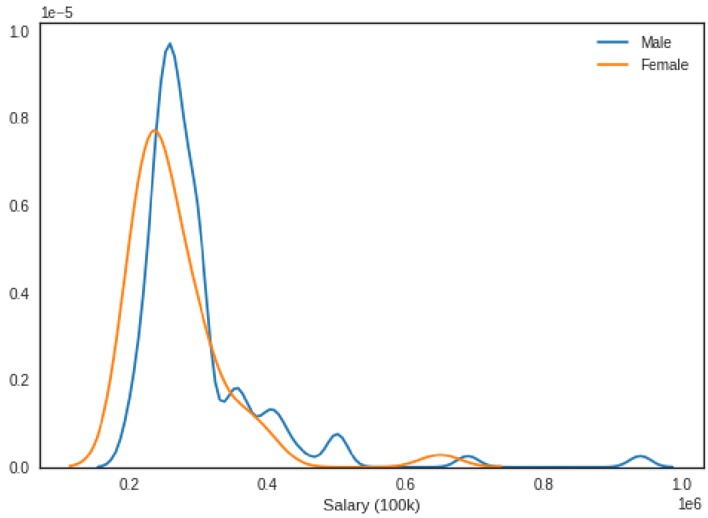
?]:
```
M    139
F     76
Name: gender, dtype: int64
```

3]:
```
sns.countplot("gender", hue="status", data=data)
plt.show()
```

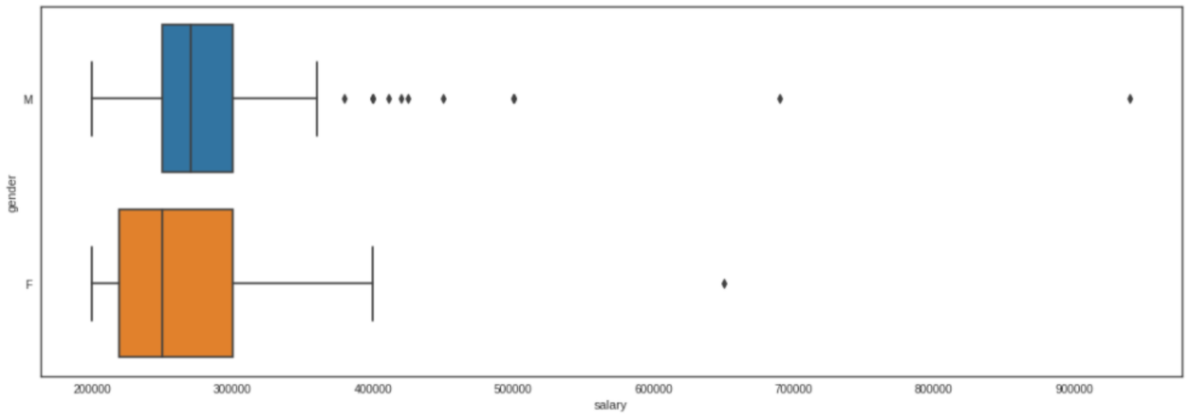


]:
```
#This plot ignores NaN values for salary, igoring students who are not placed
sns.kdeplot(data.salary[ data.gender=="M"])
sns.kdeplot(data.salary[ data.gender=="F"])
plt.legend(["Male", "Female"])
plt.xlabel("Salary (100k)")
plt.show()
```

In [99]:
```
#This plot ignores NaN values for salary, igoring students who are not placed
sns.kdeplot(data.salary[ data.gender=="M"])
sns.kdeplot(data.salary[ data.gender=="F"])
plt.legend(["Male", "Female"])
plt.xlabel("Salary (100k)")
plt.show()
```



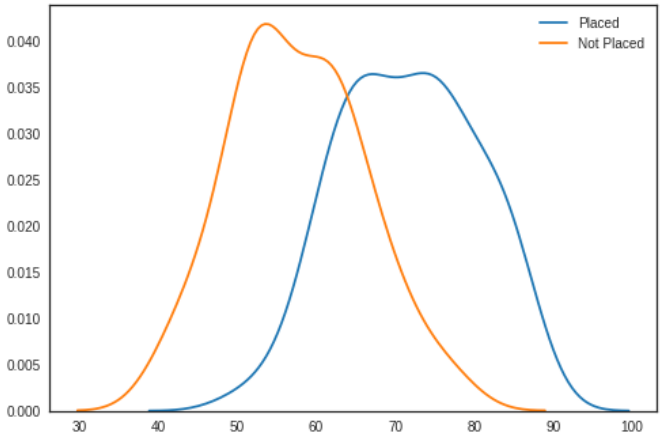
In [100]:
```
plt.figure(figsize =(18,6))
sns.boxplot("salary", "gender", data=data)
plt.show()
```



Feature: SSC_P (Secondary Education percentage), SSC_B (Board Of Education)

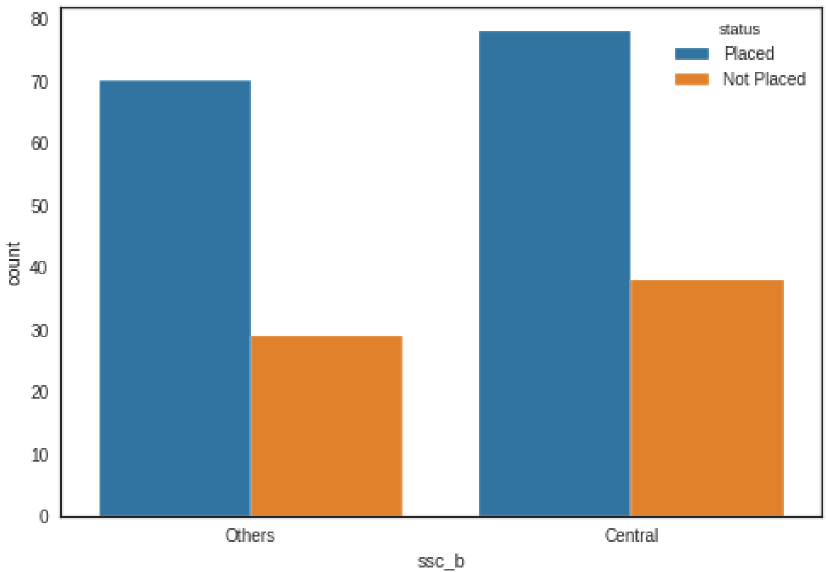
Does Secondary Education affect placements?

In [101]:
```
#Kernel-Density Plot
sns.kdeplot(data.ssc_p[ data.status=="Placed"])
sns.kdeplot(data.ssc_p[ data.status=="Not Placed"])
plt.legend(["Placed", "Not Placed"])
plt.xlabel("Secondary Education Percentage")
plt.show()
```



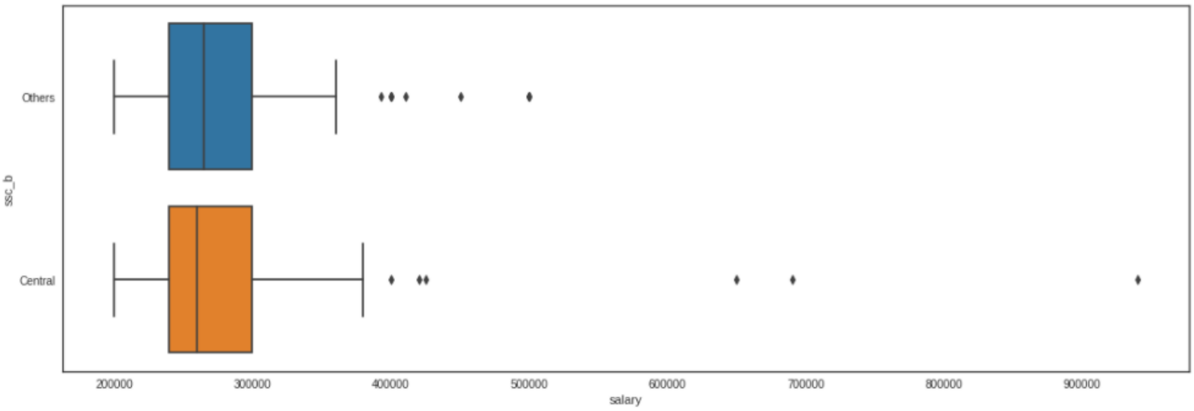
- All students with Secondary Education Percentage above 90% are placed
- All students with Secondary Education Percentage below 50% are not-placed
- **Students with good Secondary Education Percentage are placed on average.**

In [102]:
```
sns.countplot("ssc_b", hue="status", data=data)
plt.show()
```



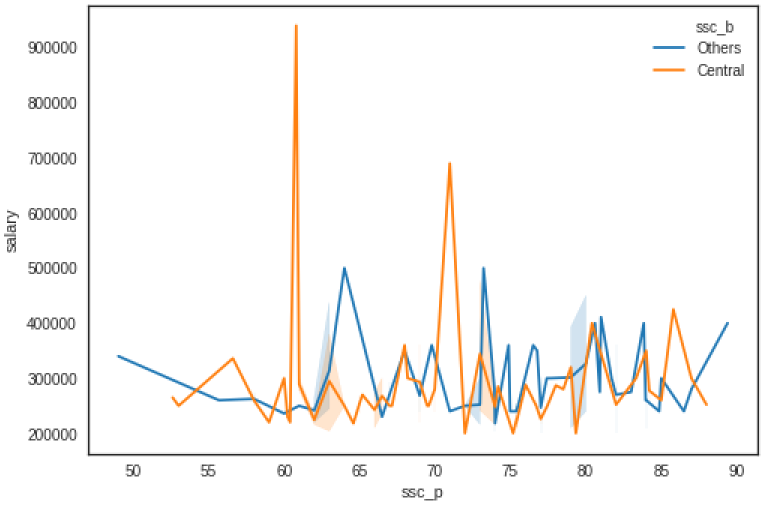
- Board Of Education does not affect Placement Status much

In [103]:
```
plt.figure(figsize =(18,6))
sns.boxplot("salary", "ssc_b", data=data)
plt.show()
```



- Outliers on both, but students from Central Board are getting the highly paid jobs.

In [104]:
```
sns.lineplot("ssc_p", "salary", hue="ssc_b", data=data)
plt.show()
```

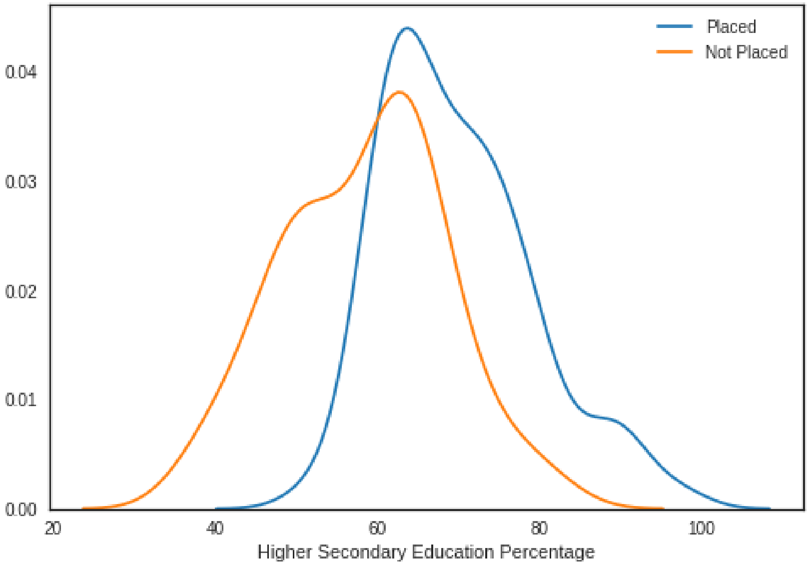


- No specific pattern (correlation) between Secondary Education Percentage and Salary.
- Board of Education is Not Affecting Salary

Feature: HSC_P (Higher Secondary Education percentage), HSC_B (Board Of Education), HSC_S (Specialization in Higher Secondary Education)

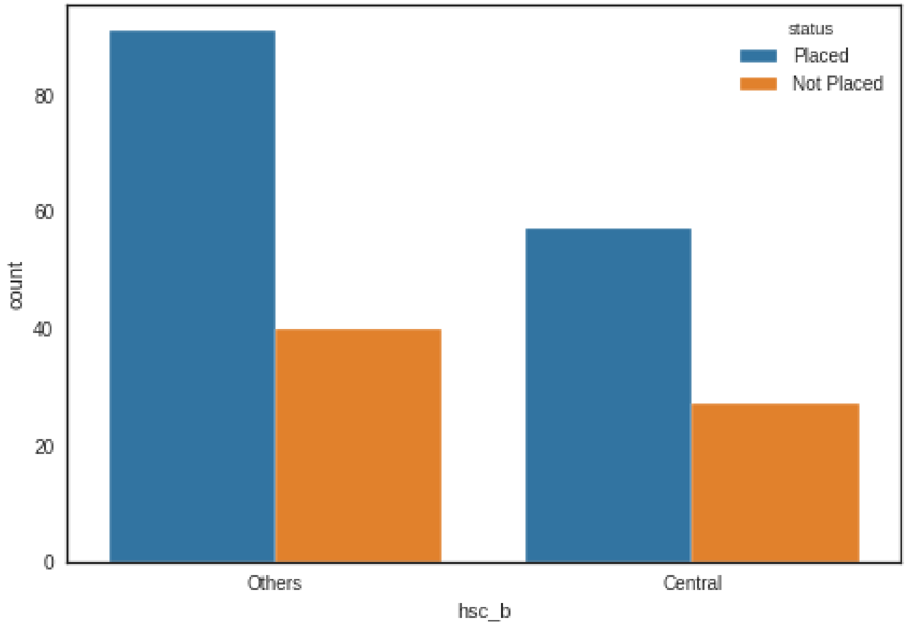
Does Higher Secondary School affect Placements?

In [105]:
```
#Kernel-Density Plot
sns.kdeplot(data.hsc_p[ data.status=="Placed"])
sns.kdeplot(data.hsc_p[ data.status=="Not Placed"])
plt.legend(["Placed", "Not Placed"])
plt.xlabel("Higher Secondary Education Percentage")
plt.show()
```



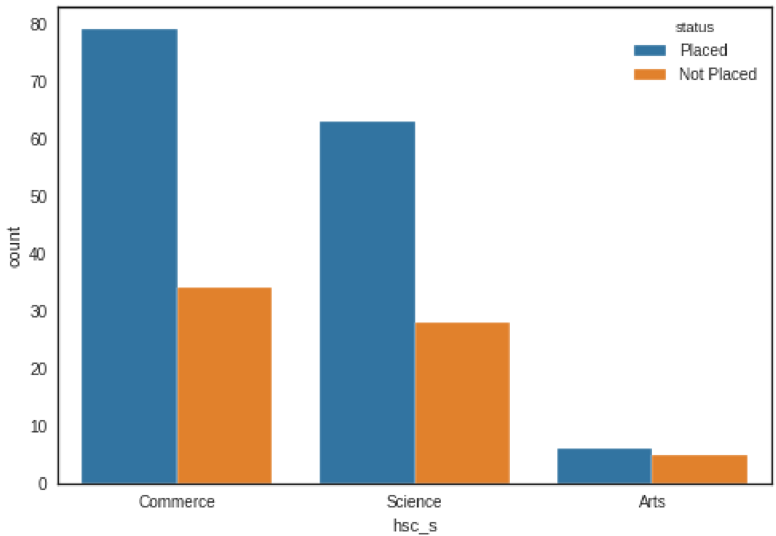
- Overlap here too. More placements for percentage above 65%
- Straight drop below 60 in placements -> Perntage must be atleast 60 for chance of being placed

In [106]:
```
sns.countplot("hsc_b", hue="status", data=data)
plt.show()
```



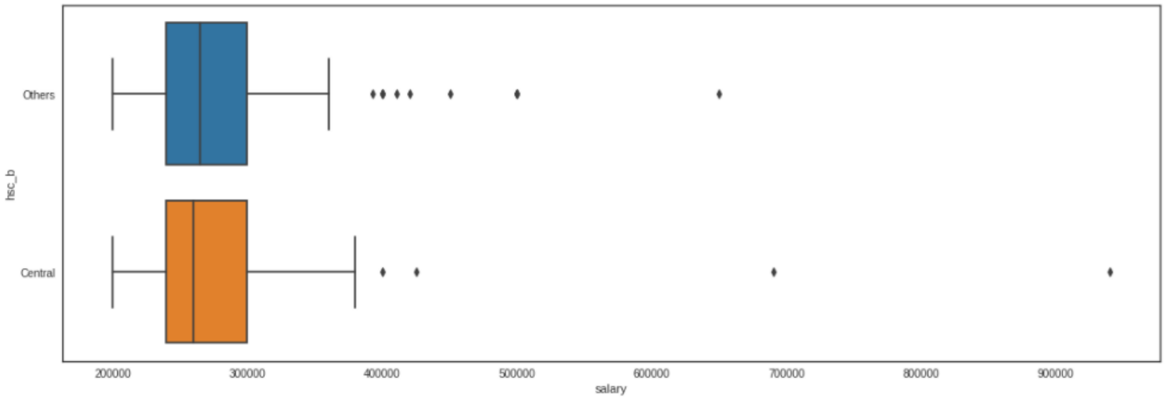
Education Board again, doesn't affect placement status much

In [107]:
```
sns.countplot("hsc_s", hue="status", data=data)
plt.show()
```



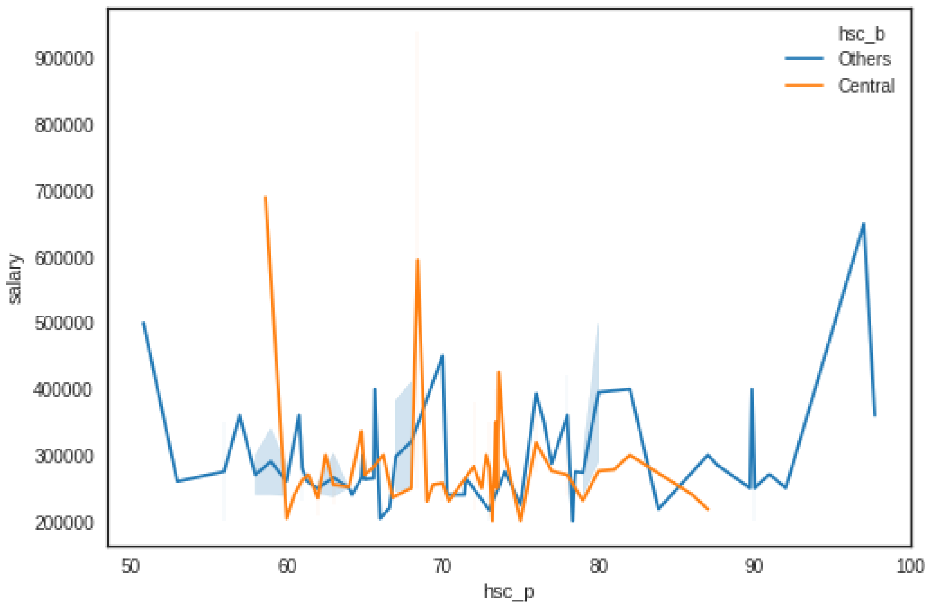
- We have very less students with Arts specialization.
- Around 2:1 placed:unplaced ratio for both Science and Commerse students

In [108]:
```
plt.figure(figsize =(18,6))
sns.boxplot("salary", "hsc_b", data=data)
plt.show()
```



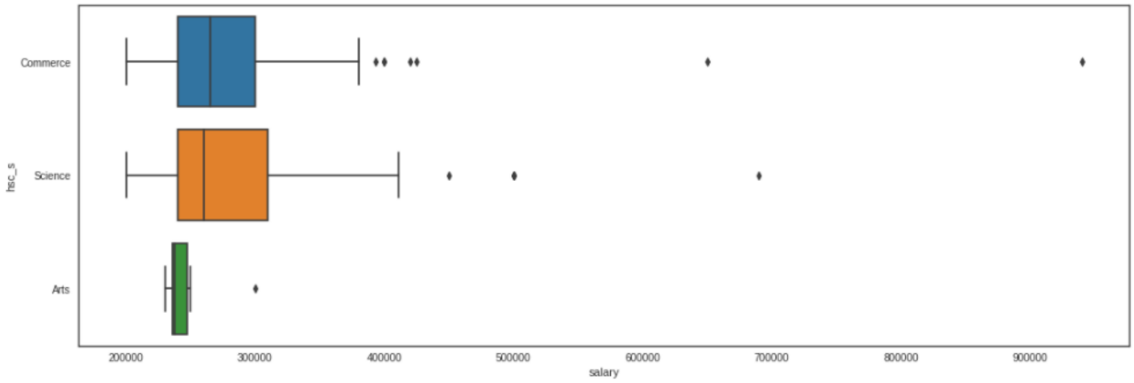
- Outliers on both, board doesn't affect getting highly paid jobs. Highest paid job was obtailed by student from Central Board though.

In [109]:
```
sns.lineplot("hsc_p", "salary", hue="hsc_b", data=data)
plt.show()
```



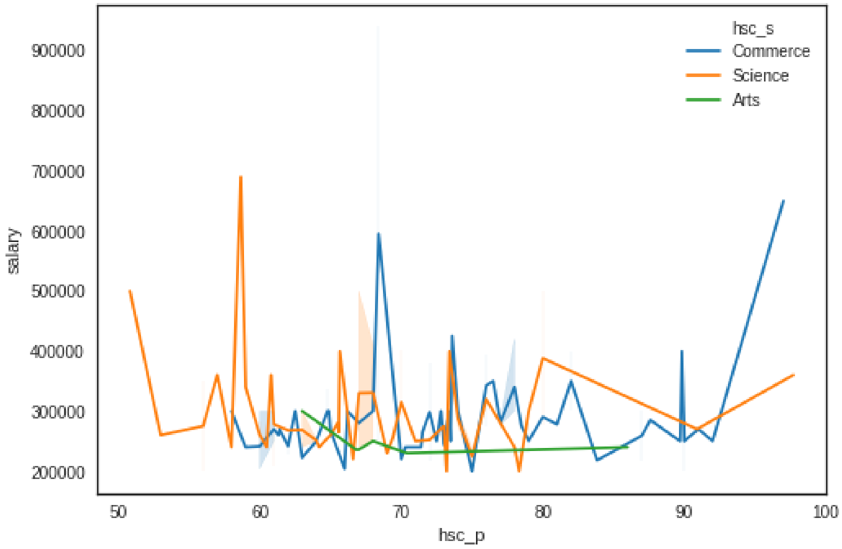
- High salary from both Central and Other.
- High salary for both high and low percentage.
- Thus, both these feature doesnot affect salary.

In [110]:
```
plt.figure(figsize =(18,6))
sns.boxplot("salary", "hsc_s", data=data)
plt.show()
```



- We can't really say for sure due to only few samples of students with Arts Major, but they aren't getting good salaries.
- Commerse students have slightly better placement status.

In [111]:
```
sns.lineplot("hsc_p", "salary", hue="hsc_s", data=data)
plt.show()
```

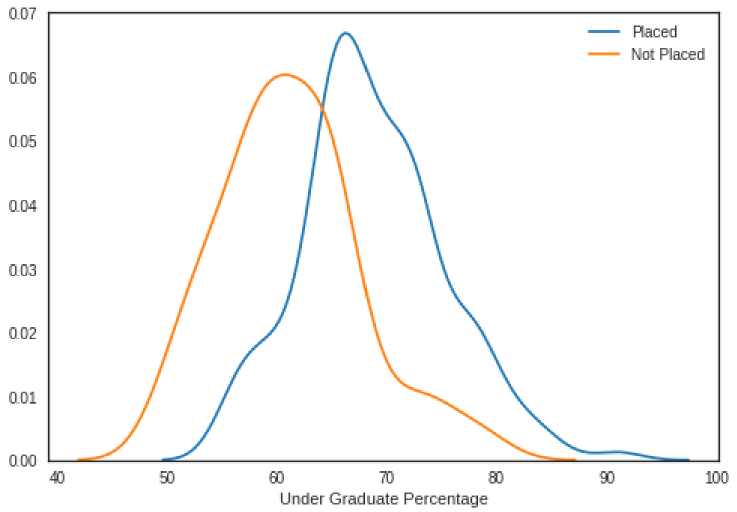


- **Student with Art Specialization surprisingly have comparatively low salary**

Feature: degree_p (Degree Percentage), degree_t (Under Graduation Degree Field)

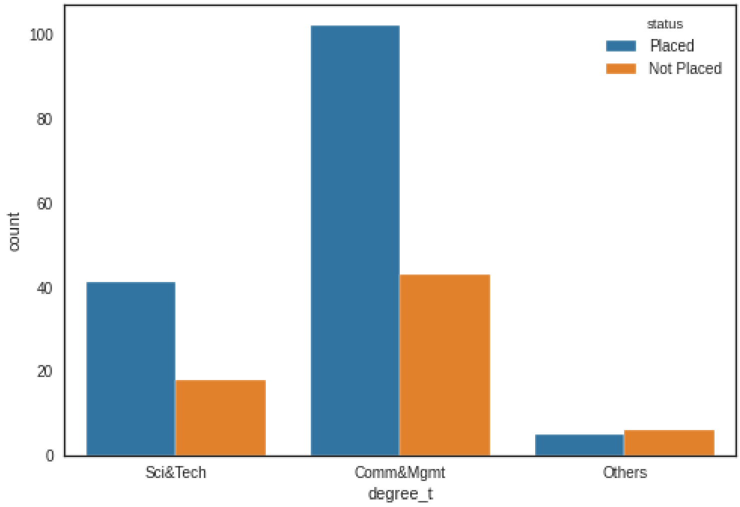
Does Under Graduate affect placements?

In [112]:
```
#Kernel-Density Plot
sns.kdeplot(data.degree_p[ data.status=="Placed"])
sns.kdeplot(data.degree_p[ data.status=="Not Placed"])
plt.legend(["Placed", "Not Placed"])
plt.xlabel("Under Graduate Percentage")
plt.show()
```



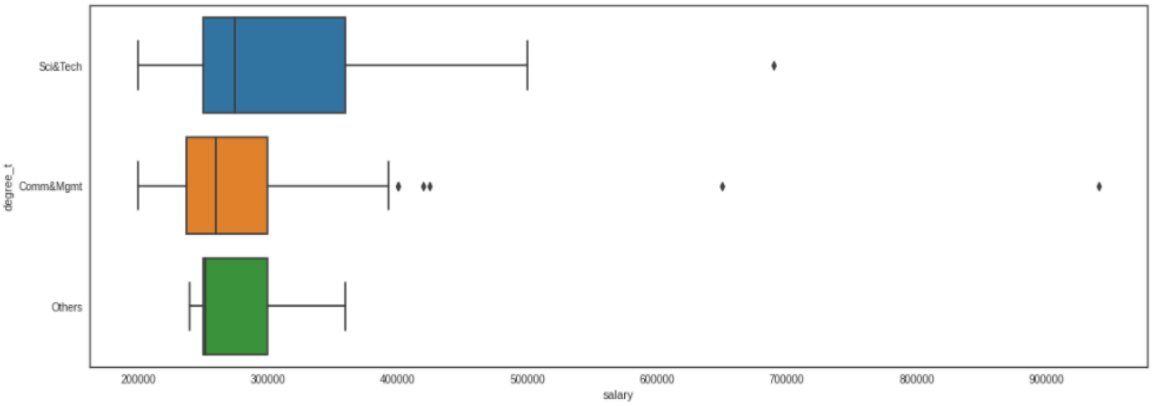
- Overlap here too. But More placements for percentage above 65.
- UG Percentage least 50% to get placement

```
In [113]: sns.countplot("degree_t", hue="status", data=data)
plt.show()
```



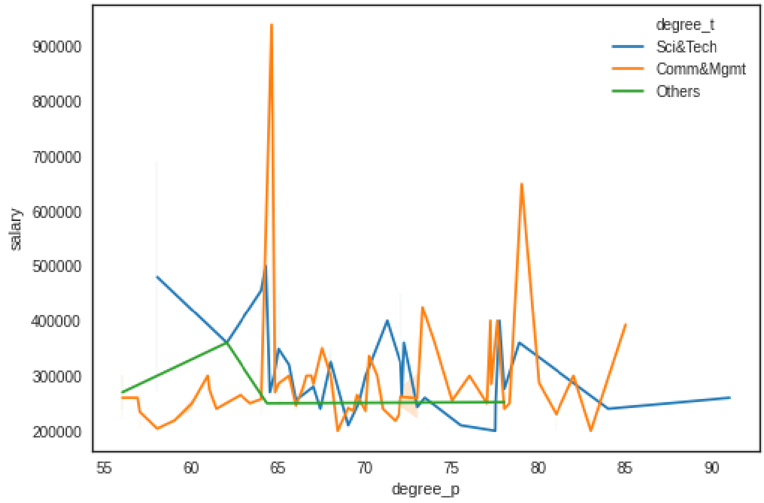
- We have very less students with "Other". We cant make decision from few cases.
- Around 2:1 placed:unplaced ratio for both Science and Commerse students

```
In [114]: plt.figure(figsize =(18,6))
sns.boxplot("salary", "degree_t", data=data)
plt.show()
```



- Science&Tech students getting more salary on average
- Management stidents are getting more highly paid dream jobs.

In [115]:
```
sns.lineplot("degree_p", "salary", hue="degree_t", data=data)
plt.show()
```

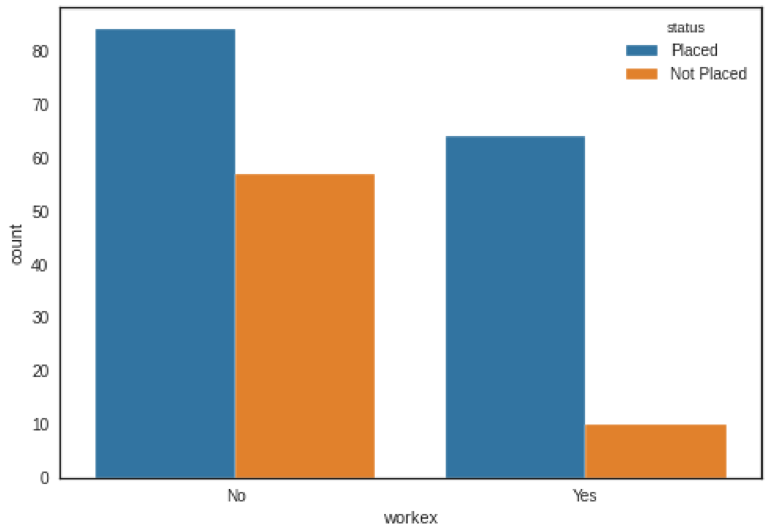


- Percentage does not seem to affect salary.
- Commerce&Mgmt students occasionally get dream placements with high salary

Feature: Workex (Work Experience)

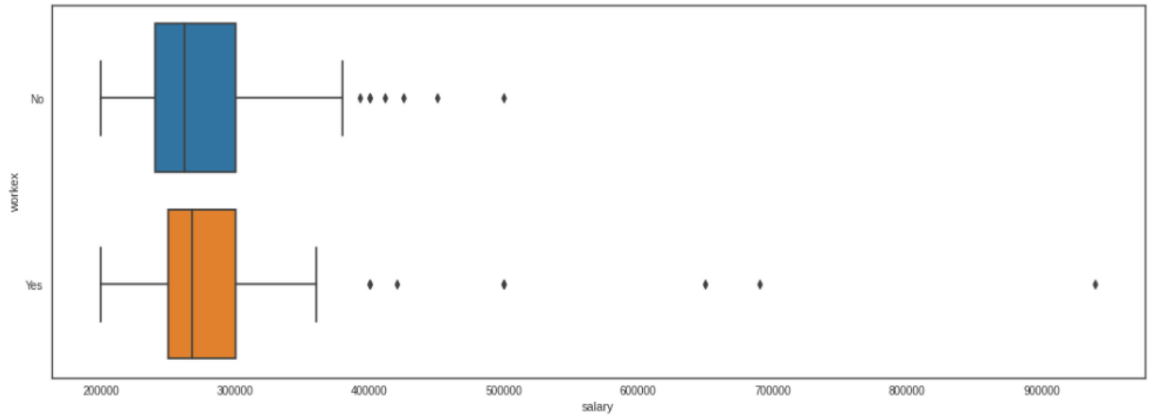
Does Work Experience affect placements?

In [116]:
```
sns.countplot("workex", hue="status", data=data)
plt.show()
```



- **This affects Placement.** Very few students with work experience not getting placed!

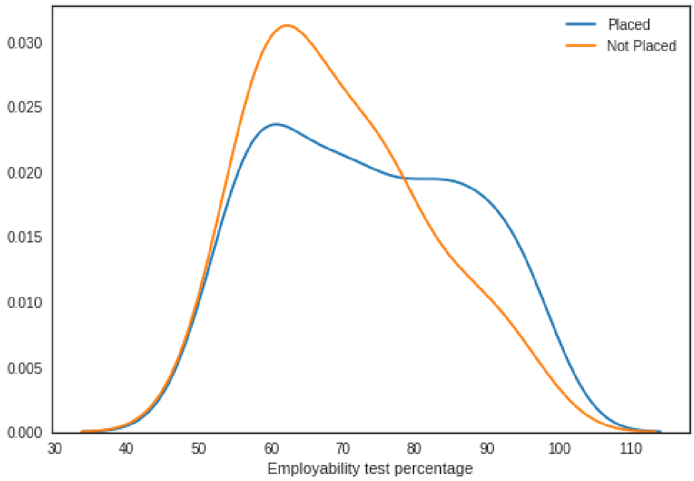
In [117]:
```
plt.figure(figsize =(18,6))
sns.boxplot("salary", "workex", data=data)
plt.show()
```



- Outliers (High salary than average) on bith end but **students with experience getting dream jobs**
- Average salary as well as base salary high for students with work experience

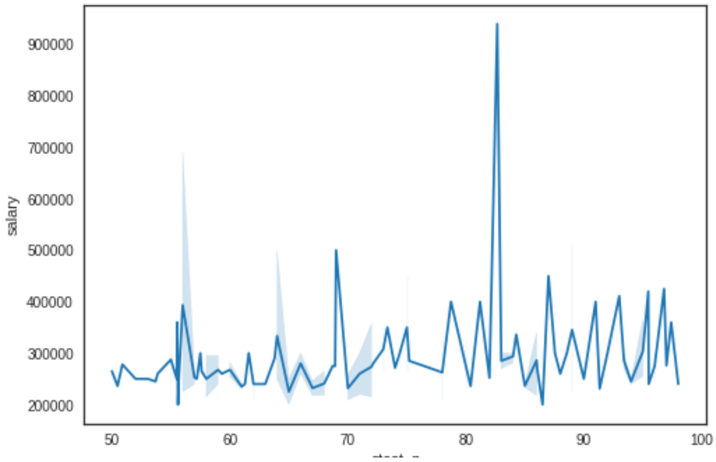
Feature: etest_p (Employability test percentage)

In [118]:
```
#Kernel-Density Plot
sns.kdeplot(data.etest_p[ data.status=="Placed"])
sns.kdeplot(data.etest_p[ data.status=="Not Placed"])
plt.legend(["Placed", "Not Placed"])
plt.xlabel("Employability test percentage")
plt.show()
```



- High overlap -> It does not affect placement status much
- More "Not Placed" on percentage 50-70 range and more placed on 80% percentage range

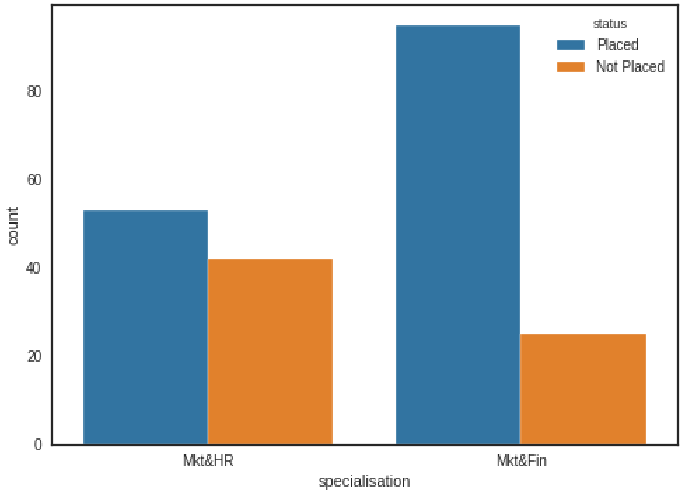
In [119]:
```
sns.lineplot("etest_p", "salary", data=data)
plt.show()
```



This feature surprisingly does not affect placements and salary much

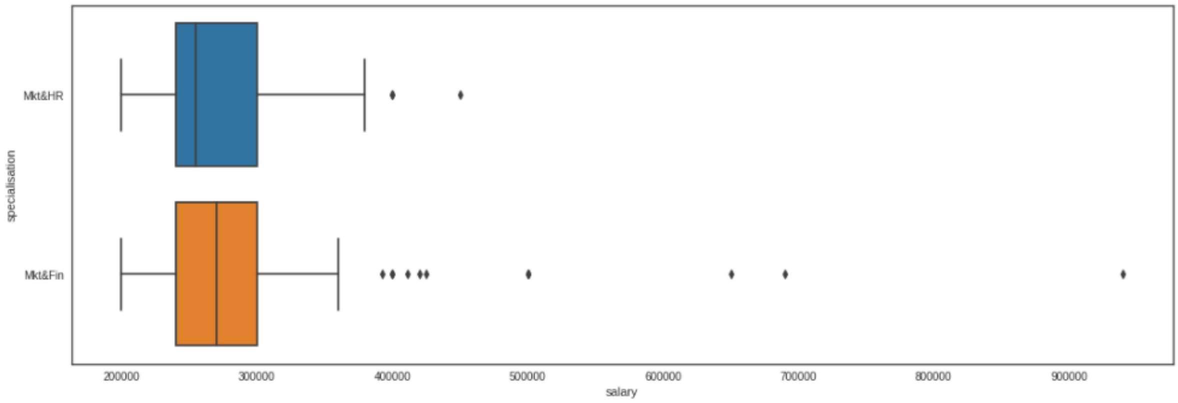
Feature: Specialisation (Post Graduate Specialization)

In [120]:
```
sns.countplot("specialisation", hue="status", data=data)
plt.show()
```



- This feature affects Placement status.
- Comparitively very low not-placed students in Mkt&Fin Section

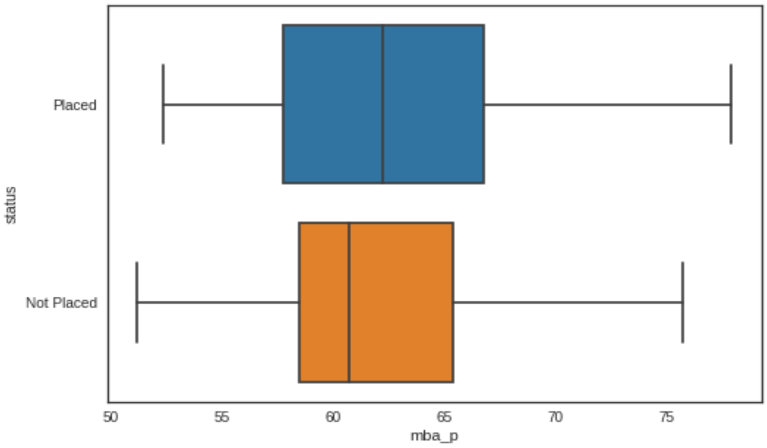
In [121]:
```
plt.figure(figsize =(18,6))
sns.boxplot("salary", "specialisation", data=data)
plt.show()
```



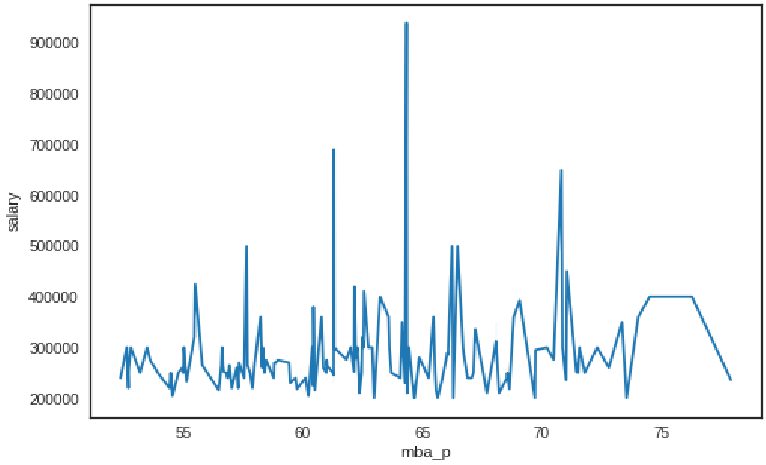
- *More Highly Paid Jobs for Mkt&Fin students *

Does MBA Percentage affect placements?

In [122]:
```
sns.boxplot("mba_p", "status", data=data)
plt.show()
```



In [123]:
```
sns.lineplot("mba_p", "salary", data=data)
plt.show()
```



MBA Percentage also deos not affect salary much

Feature Selection

Using Only following features (Ignoring Board of Education -> they didn't seem to have much effect)

- Gender
- Secondary Education percentage
- Higher Secondary Education Percentage
- Specialization in Higher Secondary Education
- Undergraduate Degree Percentage
- Under Graduation Degree Field
- Work Experience
- Employability test percentage
- Specialization
- MBA Percentage

Will compute feature importance later on.

Data Pre-Processing

In [124]:
```
data.drop(['ssc_b','hsc_b'], axis=1, inplace=True)
```

Feature Encoding

In [125]:
```
data.dtypes
# We have to encode gender,hsc_s, degree_t, workex, specialisation and status
```

Out[125]:
```
gender             object
ssc_p             float64
hsc_p             float64
hsc_s              object
degree_p          float64
degree_t           object
workex             object
etest_p           float64
specialisation     object
mba_p             float64
status             object
salary            float64
dtype: object
```

In [126]:
```
data["gender"] = data.gender.map({"M":0,"F":1})
data["hsc_s"] = data.hsc_s.map({"Commerce":0,"Science":1,"Arts":2})
data["degree_t"] = data.degree_t.map({"Comm&Mgmt":0,"Sci&Tech":1, "Others":2})
data["workex"] = data.workex.map({"No":0, "Yes":1})
data["status"] = data.status.map({"Not Placed":0, "Placed":1})
data["specialisation"] = data.specialisation.map({"Mkt&HR":0, "Mkt&Fin":1})
```

Problem Statement

- Predicting If Students gets placed or not (Binary Classification Problem)
- Predicting Salary of Student (Regression Problem)

In [127]:
```
#Lets make a copy of data, before we proceeed with specific problems
data_clf = data.copy()
data_reg = data.copy()
```

Binary Classification Problem

Decision Tree Based Models

Using Decision Tree based Algorithm does not require feature scaling, and works great also in presence of categorical columns without ONE_HOT Encoding

In [128]:
```
# Library imports
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

Dropping Salary Feature

Filling 0s for salary of students who didn't get placements would be bad idea as it would mean student gets placement if he earns salary.

In [129]:
```
# Seperating Features and Target
X = data_clf[['gender', 'ssc_p', 'hsc_p', 'hsc_s', 'degree_p', 'degree_t', 'workex','etest_p', 'specialisation', 'mba_p',]]
y = data_clf['status']
```

In [130]:
```
#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [131]:
```
dtree = DecisionTreeClassifier(criterion='entropy')
dtree.fit(X_train, y_train)
y_pred = dtree.predict(X_test)
```

In [132]:
```
accuracy_score(y_test, y_pred)
```

Out[132]:
```
0.8307692307692308
```

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

```
              precision    recall  f1-score   support

           0       0.65      0.76      0.70        17
           1       0.91      0.85      0.88        48

    accuracy                           0.83        65
   macro avg       0.78      0.81      0.79        65
weighted avg       0.84      0.83      0.83        65
```

In [134]:
```
#Using Random Forest Algorithm
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, y_train)
y_pred = random_forest.predict(X_test)
```

In [135]:
```
accuracy_score(y_test, y_pred)
```

Out[135]:
```
0.9230769230769231
```

In [136]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85        17
           1       0.94      0.96      0.95        48

    accuracy                           0.92        65
   macro avg       0.91      0.89      0.90        65
```

```
weighted avg       0.92      0.92      0.92        65
```

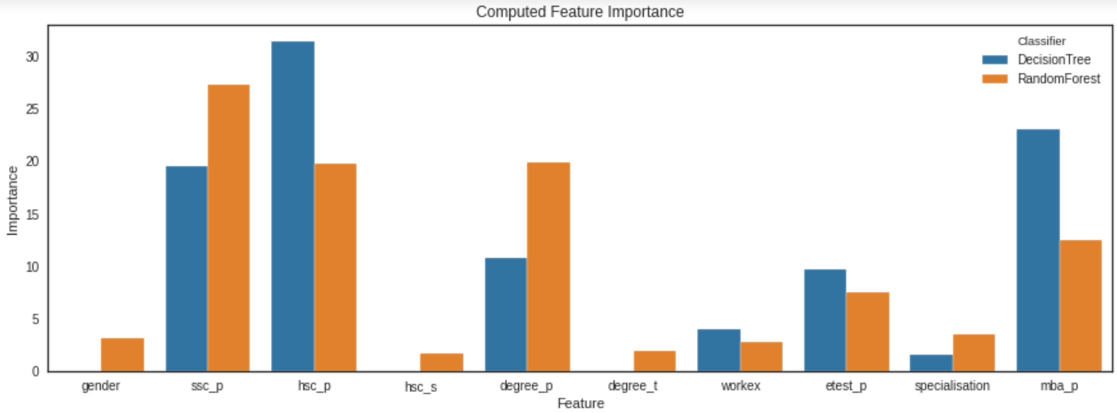
Feature Importance (Percentage)

Tree based algorithms can be used to compute feature importance

Checking feature importance obtained from these:

In [137]:
```
rows = list(X.columns)
imp = pd.DataFrame(np.zeros(6*len(rows)).reshape(2*len(rows), 3))
imp.columns = ["Classifier", "Feature", "Importance"]
#Add Rows
for index in range(0, 2*len(rows), 2):
    imp.iloc[index] = ["DecisionTree", rows[index//2], (100*dtree.feature_importances_[index//2])]
    imp.iloc[index + 1] = ["RandomForest", rows[index//2], (100*random_forest.feature_importances_[index//2])]
```

In [138]:
```
plt.figure(figsize=(15,5))
sns.barplot("Feature", "Importance", hue="Classifier", data=imp)
plt.title("Computed Feature Importance")
plt.show()
```



hsc_s -> Specialization in Higher Secondary Education

degree_t -> Under Graduation(Degree type)- Field of degree education

specialisation -> Post Graduation(MBA)- Specialization

Field of study does not seem to affect much

Optionally we can remove these least important features and re-clssify data.

Binary Classification with Logistic Regression

One Hot Encoding

Encoding Categorical Features

In [139]:
```
# Seperating Features and Target
X = data_clf[['gender', 'ssc_p', 'hsc_p', 'hsc_s', 'degree_p', 'degree_t', 'workex','etest_p', 'specialisation', 'mba_p',]]
y = data_clf['status']
#Reverse Mapping and making Categorical
X["gender"] = pd.Categorical(X.gender.map({0:"M",1:"F"}))
X["hsc_s"] = pd.Categorical(X.hsc_s.map({0:"Commerce",1:"Science",2:"Arts"}))
X["degree_t"] = pd.Categorical(X.degree_t.map({0:"Comm&Mgmt",1:"Sci&Tech",2:"Others"}))
X["workex"] = pd.Categorical(X.workex.map({0:"No",1:"Yes"}))
X["specialisation"] = pd.Categorical(X.specialisation.map({0:"Mkt&HR",1:"Mkt&Fin"}))
```

In [140]:
```
#One-Hot Encoding
X = pd.get_dummies(X)
colmunn_names = X.columns.to_list()
```

Feature Scaling

- Percentages are on scale 0-100
- Categorical Features are on range 0-1 (By one hot encoding)
- High Scale for Salary -> Salary is heavily skewed too -> SkLearn has RobustScaler which might work well here

Scaling Everything between 0 and 1 (This wont affect one-hot encoded values)

In [141]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

In [142]:
```
#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3)
```

In [143]:
```
from sklearn.linear_model import LogisticRegression
```

In [144]:
```
logistic_reg = LogisticRegression()
logistic_reg.fit(X_train, y_train)
y_pred = logistic_reg.predict(X_test)
```

In [145]:
```
accuracy_score(y_test, y_pred)
```

Out[145]:
```
0.8
```

In [146]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.65      0.70        23
           1       0.82      0.88      0.85        42

    accuracy                           0.80        65
   macro avg       0.79      0.77      0.77        65
weighted avg       0.80      0.80      0.80        65
```

Computating Feature importance by Mean Decrease Accuracy (MDA)

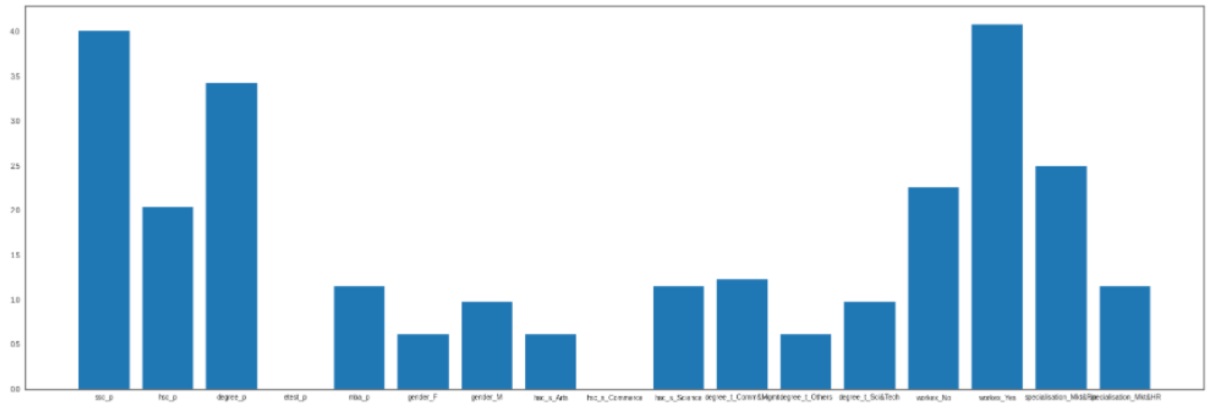
Since Logistic Regression performed well, Lets run another method for determining fearure importance here.

```
In [147]: import eli5
          from eli5.sklearn import PermutationImportance
          perm = PermutationImportance(logistic_reg).fit(X_test, y_test)
          eli5.show_weights(perm)
```

Out[147]:

Weight	Feature
0.1231 ± 0.0802	x0
0.0492 ± 0.0685	x2
0.0431 ± 0.0408	x1
0.0400 ± 0.0816	x14
0.0154 ± 0.0195	x12
0.0031 ± 0.0452	x13
0 ± 0.0000	x3
0 ± 0.0000	x8
-0.0031 ± 0.0123	x7
-0.0031 ± 0.0123	x11
-0.0092 ± 0.0246	x10
-0.0123 ± 0.0123	x5
-0.0123 ± 0.0230	x4
-0.0154 ± 0.0195	x6
-0.0185 ± 0.0230	x9
-0.0185 ± 0.0230	x16
-0.0215 ± 0.0500	x15

```
In [148]: plt.figure(figsize=(30, 10))
          plt.bar(colmunn_names , perm.feature_importances_std_ * 100)
          plt.show()
```



From Feature Importance of Tree-based Algorithms and MDA we can conclude that:

- Academic performance affects placement (All percentages had importantance)
- Work Experience Effects Placement
- Gender and Specialization in Commerse (in higher-seondary and undergraduate) also has effect on placements.

Prediction of Salary (Regression Analysis)

In [149]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from sklearn.metrics import mean_absolute_error, r2_score
```

Data Preprocessing

In [150]:
```
#dropping NaNs (in Salary)
data_reg.dropna(inplace=True)
#dropping Status = "Placed" column
data_reg.drop("status", axis=1, inplace=True)
```

In [151]:
```
data_reg.head()
```

Out[151]:

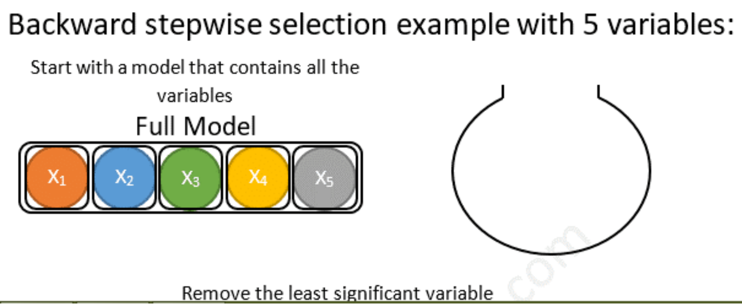
	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	salary
0	0	67.00	91.00	0	58.00	1	0	55.0	0	58.80	270000.0
1	0	79.33	78.33	1	77.48	1	1	86.5	1	66.28	200000.0
2	0	65.00	68.00	2	64.00	0	0	75.0	1	57.80	250000.0
4	0	85.80	73.60	0	73.30	0	0	96.8	1	55.50	425000.0
7	0	82.00	64.00	1	66.00	1	1	67.0	1	62.14	252000.0

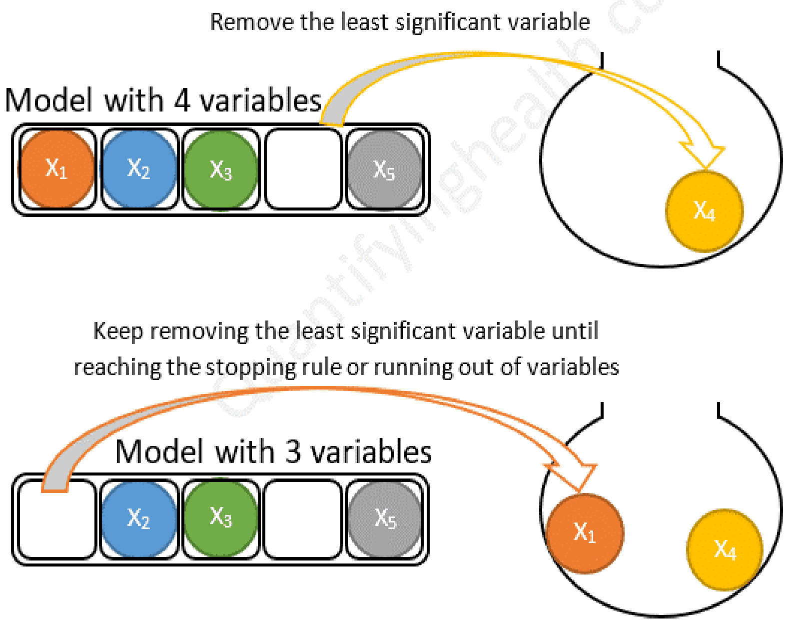
In [152]:
```
#Seperating Depencent and Independent Vaiiables
y = data_reg["salary"] #Dependent Variable
X = data_reg.drop("salary", axis=1)
column_names = X.columns.values
```

In [153]:
```
#Scalizing between 0-1 (Normalization)
X_scaled = MinMaxScaler().fit_transform(X)
```

Feature Selection

** Not all features are significant. Thus, let's perform a feature selection procedure**





Determining Least Significant Variable

The least significant variable is a variable which:

- has the highest p-value
- Removing it reduces R2 to lowest value compared to other features
- Removing it has least increment in residuals-sum-of-squares (RSS)

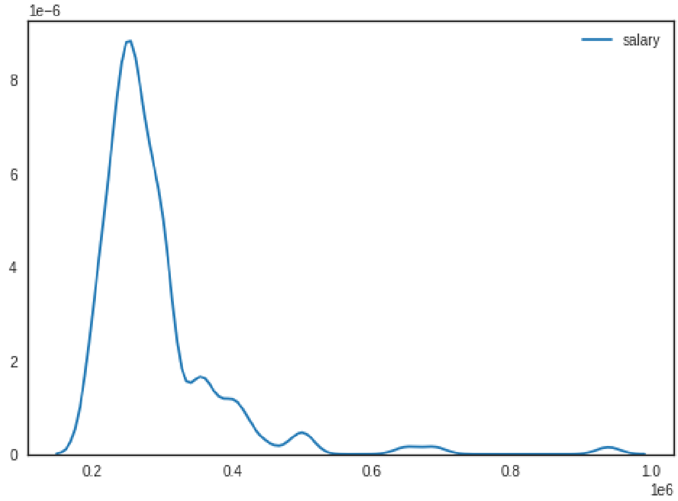
Outliers' Removal

Feature Selecton cannot perform well in presence of outliers. Lets identy and remove outliers before proceding

n [154]:

```
#PDF of Salary
sns.kdeplot(y)
plt.show()
```

In [154]:
```
#PDF of Salary
sns.kdeplot(y)
plt.show()
```



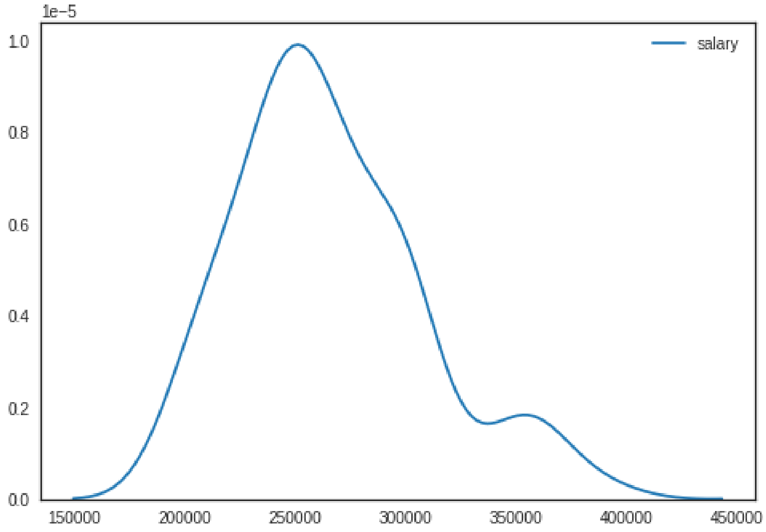
It is clear that very few students have salary greater than 400,000 (hence outliers)

In [155]:
```
#Selecting outliers
y[y > 400000]
# 9 records
```

Out[155]:
```
4      425000.0
39     411000.0
53     450000.0
77     500000.0
95     420000.0
119    940000.0
150    690000.0
163    500000.0
174    500000.0
177    650000.0
Name: salary, dtype: float64
```

In [156]:
```
#Removing these Records from data
X_scaled = X_scaled[y < 400000]
y = y[y < 400000]
```

In [157]:
```
#PDF of Salary without outliers. Still skewed though
sns.kdeplot(y)
plt.show()
```

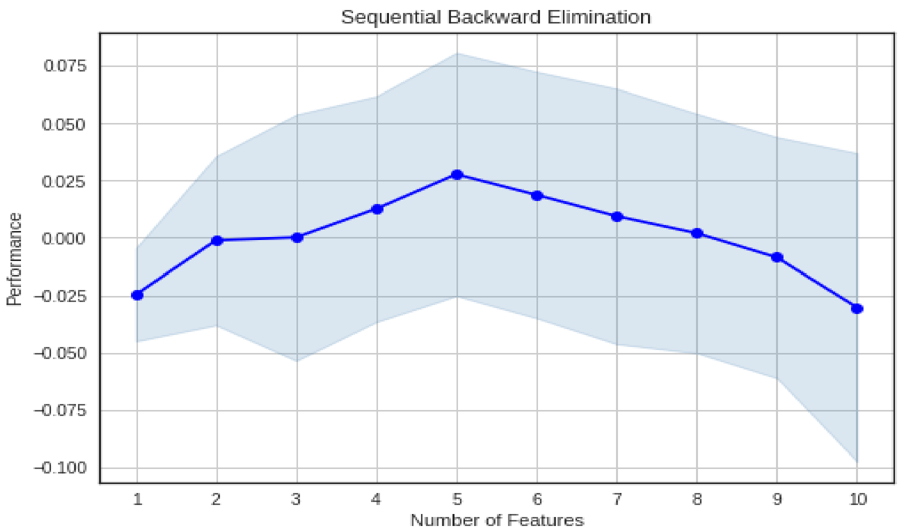


1. Determining Least Significant Variable by R2 Score

In [158]:
```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
```

In [159]:
```
linreg = LinearRegression()
sfs = SFS(linreg, k_features=1, forward=False, scoring='r2',cv=10)
sfs = sfs.fit(X_scaled, y)
fig = plot_sfs(sfs.get_metric_dict(), kind='std_err')

plt.title('Sequential Backward Elimination')
plt.grid()
plt.show()
#From Plot its clear that, many features actually decrease the performance
```



In [160]:
```python
# Lets see the top 5 most significant features
top_n = 5
sfs.get_metric_dict()[top_n]
```

Out[160]:
```
{'feature_idx': (0, 3, 5, 7, 9),
 'cv_scores': array([-0.09997564, -0.11551795,  0.14652782,  0.19241391, -0.19535134,
        -0.13235138, -0.03896556,  0.3116134 ,  0.13836643,  0.07020936]),
 'avg_score': 0.027696903219017056,
 'feature_names': ('0', '3', '5', '7', '9'),
 'ci_bound': 0.11802751969012426,
 'std_dev': 0.15891404557580263,
 'std_err': 0.05297134852526754}
```

In [161]:
```python
#Top N Features
top_n_indices = list(sfs.get_metric_dict()[top_n]['feature_idx'])
print(f"Most Significant {top_n} Features:")
for col in column_names[top_n_indices]:
    print(col)
```

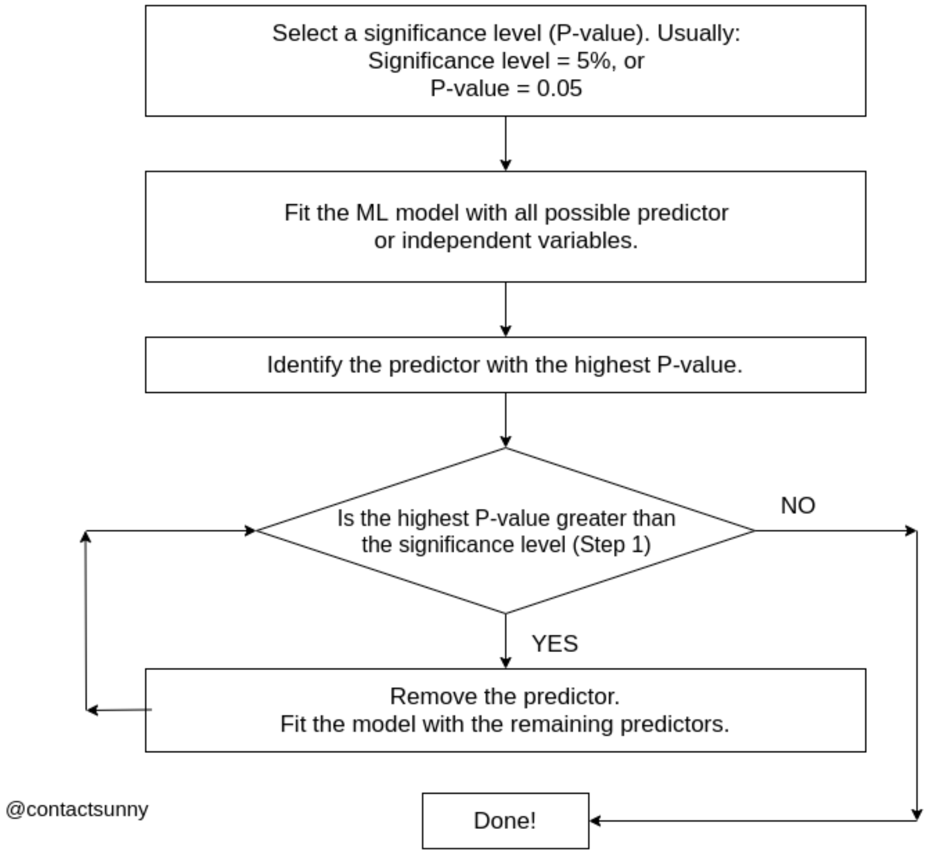
```
Most Significant 5 Features:
gender
hsc_s
degree_t
etest_p
mba_p
```

In [162]:
```python
#Select these Features only
X_selected = X_scaled[: ,top_n_indices]
lin_reg = LinearRegression()
lin_reg.fit(X_selected, y)
y_pred = lin_reg.predict(X_selected)
print(f"R2 Score: {r2_score(y, y_pred)}")
print(f"MAE: {mean_absolute_error(y, y_pred)}")
```

```
R2 Score: 0.1101660718969637
MAE: 30630.128295211573
```

This is the best I could do with Linear Regression

Determining Least Significant Variable by P-Value



In [163]:
```
#Converting to DF for as  column names gives readibility
X_scaled = pd.DataFrame(X_scaled, columns=column_names)
y = y.values

# We must add a constants 1s for intercept before doing Linear Regression with statsmodel
X_scaled = sm.add_constant(X_scaled)
X_scaled.head()
#Constants 1 added for intercept term
```

Out[163]:

	const	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p
0	1.0	0.0	0.445545	0.857051	0.0	0.057143	0.5	0.0	0.104167	0.0	0.251666
1	1.0	0.0	0.750743	0.586729	0.5	0.613714	0.5	1.0	0.760417	1.0	0.544884
2	1.0	0.0	0.396040	0.366332	1.0	0.228571	0.0	0.0	0.520833	1.0	0.212466
3	1.0	0.0	0.816832	0.280990	0.5	0.285714	0.5	1.0	0.354167	1.0	0.382595
4	1.0	0.0	0.594059	0.601024	0.0	0.457143	0.0	0.0	0.861250	1.0	0.349275

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.123
Model:	OLS	**Adj. R-squared:**	0.052
Method:	Least Squares	**F-statistic:**	1.722
Date:	Sat, 05 Jun 2021	**Prob (F-statistic):**	0.0829
Time:	05:40:18	**Log-Likelihood:**	-1608.4
No. Observations:	134	**AIC:**	3239.
Df Residuals:	123	**BIC:**	3271.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.625e+05	1.28e+04	20.498	0.000	2.37e+05	2.88e+05
gender	-1.784e+04	8299.998	-2.149	0.034	-3.43e+04	-1406.775
ssc_p	-116.6148	2.04e+04	-0.006	0.995	-4.04e+04	4.02e+04
hsc_p	-1.842e+04	2.13e+04	-0.864	0.389	-6.06e+04	2.38e+04
hsc_s	-2.775e+04	1.58e+04	-1.761	0.081	-5.9e+04	3444.983
degree_p	-9885.6991	2.25e+04	-0.438	0.662	-5.45e+04	3.47e+04
degree_t	3.947e+04	1.69e+04	2.340	0.021	6077.584	7.29e+04
workex	-7748.2212	7673.070	-1.010	0.315	-2.29e+04	7440.151
etest_p	1.839e+04	1.43e+04	1.286	0.201	-9906.447	4.67e+04
specialisation	2457.2424	8013.710	0.307	0.760	-1.34e+04	1.83e+04
mba_p	3.704e+04	2.11e+04	1.756	0.082	-4717.648	7.88e+04

Omnibus:	10.852	**Durbin-Watson:**	1.965
Prob(Omnibus):	0.004	**Jarque-Bera (JB):**	11.041
Skew:	0.661	**Prob(JB):**	0.00400
Kurtosis:	3.477	**Cond. No.**	12.9

RESULT

OLS Regression Results

Dep. Variable:	y	R-squared:	0.123
Model:	OLS	Adj. R-squared:	0.052
Method:	Least Squares	F-statistic:	1.722
Date:	Sat, 05 Jun 2021	Prob (F-statistic):	0.0829
Time:	05:40:18	Log-Likelihood:	-1608.4
No. Observations:	134	AIC:	3239.
Df Residuals:	123	BIC:	3271.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.625e+05	1.28e+04	20.498	0.000	2.37e+05	2.88e+05
gender	-1.784e+04	8299.998	-2.149	0.034	-3.43e+04	-1406.775
ssc_p	-116.6148	2.04e+04	-0.006	0.995	-4.04e+04	4.02e+04
hsc_p	-1.842e+04	2.13e+04	-0.864	0.389	-6.06e+04	2.38e+04
hsc_s	-2.775e+04	1.58e+04	-1.761	0.081	-5.9e+04	3444.983
degree_p	-9885.6991	2.25e+04	-0.438	0.662	-5.45e+04	3.47e+04
degree_t	3.947e+04	1.69e+04	2.340	0.021	6077.584	7.29e+04
workex	-7748.2212	7673.070	-1.010	0.315	-2.29e+04	7440.151
etest_p	1.839e+04	1.43e+04	1.286	0.201	-9906.447	4.67e+04
specialisation	2457.2424	8013.710	0.307	0.760	-1.34e+04	1.83e+04
mba_p	3.704e+04	2.11e+04	1.756	0.082	-4717.648	7.88e+04

Omnibus:	10.852	Durbin-Watson:	1.965
Prob(Omnibus):	0.004	Jarque-Bera (JB):	11.041
Skew:	0.661	Prob(JB):	0.00400
Kurtosis:	3.477	Cond. No.	12.9

In [165]:

```
# Identify max P-value (P>|t|) column
# Feature ssc_p has 0.995
#drop ssc_p
X_scaled = X_scaled.drop('ssc_p', axis=1)
model = sm.OLS(y, X_scaled)
results = model.fit()
results.summary()
```

Out[165]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.123
Model:	OLS	**Adj. R-squared:**	0.059
Method:	Least Squares	**F-statistic:**	1.929
Date:	Sat, 05 Jun 2021	**Prob (F-statistic):**	0.0536
Time:	05:40:18	**Log-Likelihood:**	-1608.4
No. Observations:	134	**AIC:**	3237.
Df Residuals:	124	**BIC:**	3266.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.625e+05	1.2e+04	21.888	0.000	2.39e+05	2.86e+05
gender	-1.784e+04	8177.285	-2.182	0.031	-3.4e+04	-1657.933
hsc_p	-1.845e+04	2.08e+04	-0.888	0.376	-5.96e+04	2.27e+04

Dep. Variable:	y	**R-squared:**	0.079
Model:	OLS	**Adj. R-squared:**	0.057
Method:	Least Squares	**F-statistic:**	3.703
Date:	Sat, 05 Jun 2021	**Prob (F-statistic):**	0.0135
Time:	05:40:18	**Log-Likelihood:**	-1611.7
No. Observations:	134	**AIC:**	3231.
Df Residuals:	130	**BIC:**	3243.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.573e+05	7260.622	35.441	0.000	2.43e+05	2.72e+05
gender	-2.036e+04	7777.446	-2.618	0.010	-3.57e+04	-4973.494
degree_t	2.202e+04	1.33e+04	1.654	0.101	-4325.122	4.84e+04
mba_p	3.138e+04	1.67e+04	1.884	0.062	-1567.309	6.43e+04

Omnibus:	13.932	**Durbin-Watson:**	2.008
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	14.967
Skew:	0.773	**Prob(JB):**	0.000562
Kurtosis:	3.539	**Cond. No.**	5.77

CONCLUSION

From the machine learning models we identifies what are all the features that affects the salary.
Therefore the top 5 feature that affects the salary are:

- gender -> Gender
- degree_t -> Under Graduation(Degree type)- Field of degree education
- mba_p -> MBA percentage
- hsc_s -> Specialization in Higher Secondary Education
- etest_p -> Employability test percentage

REFERENCES

- https://www.kaggle.com/albertonavaa/beginner-approach-to-campus-placement-prediction
- kaggle.com/cs155925/campus-recruitement-exploration
- https://www.kaggle.com/srikardornala/placement-dataset-regression-classification
- https://www.kaggle.com/nareshbhat/outlier-the-silent-killer
- https://www.kaggle.com/benroshan/you-re-hired-analysis-on-campus-recruitment-data