Hope Artificial Intelligence

Inferential Analysis

Problem Statement or Requirement:

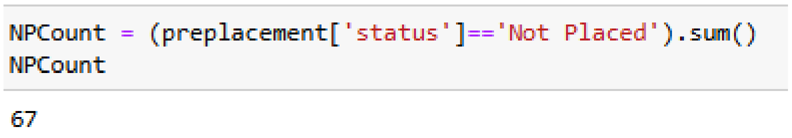
---

1)Replace the NaN values with correct value. And justify why you have chosen the same.

I have taken mean value rather than zero. Since my prediction aims at salary. Zero value will tell our model that a candidate with certain academic results is likely to earn Zero. This will skew the model's predictions towards lower salaries for all candidates.

---

2)How many of them are not placed?



```
NPCount = (preplacement['status']=='Not Placed').sum()
NPCount
```

```
67
```

67 candidates are not placed.

---

3)Find the reason for non-placement from the dataset?

Finding the reason for **non-placement** requires a **Predictive Modeling** or **Classification** approach using dataset. We need to identify which factors (columns) significantly distinguish between the students who were placed and those who were not.

Here is a plan using common data science technique in Python:

1. Prepare the Data

- **Target Variable:** 'status' where 'Placed' is one value and 'Not Placed'
- **Feature Selection:** Selected all relevant predictor columns (e.g., ssc_p, hsc_p, degree_p, gender, specialization, etc.).

Example Code (using one-hot encoding for categorical data):

Python

Hope Artificial Intelligence

Inferential Analysis

```
In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split
```

```
In [2]: dataset = pd.read_csv('Placement.csv')
```

```
In [3]: df = dataset.copy()
        # Convert the target to a binary number (e.g., 1 for Placed, 0 for Not Placed)
        df['is_placed'] = df['status'].apply(lambda x: 1 if x == 'Placed' else 0)
        # Select features (excluding the original 'status' and any ID columns)
        X = df.drop(['status', 'is_placed'], axis=1)
        y = df['is_placed']
        # Convert categorical variables into dummy/indicator variables (required for most models)
        X = pd.get_dummies(X, drop_first=True)

        # Handle missing values (important step! Fill with median/mode or drop rows)
        X = X.fillna(X.median())
```

```
In [4]: X.isna().sum()
```

2. Use a Classification Model to Rank Factors

The most effective way to determine the **"reasons"** is by using a model that provides **Feature Importance Scores**. **Decision Trees** and **Random Forests** are ideal for this.

Python

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Split data for training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Get feature importances
importance = pd.Series(model.feature_importances_, index=X_train.columns)
```

```
C:\Users\Ilango\AppData\Local\anaconda3\envs\aiml\lib\site-packages\sklearn\utils\fixes.py:230: DeprecationWarning: distutils V
ersion classes are deprecated. Use packaging.version instead.
  if _joblib.__version__ >= LooseVersion('0.12'):
C:\Users\Ilango\AppData\Local\anaconda3\envs\aiml\lib\site-packages\sklearn\utils\fixes.py:230: DeprecationWarning: distutils V
ersion classes are deprecated. Use packaging.version instead.
  if _joblib.__version__ >= LooseVersion('0.12'):
```

3. Identify and Interpret the Top Reasons

The factors with the **highest importance scores** are the most predictive of placement status.

Python

Inferential Analysis

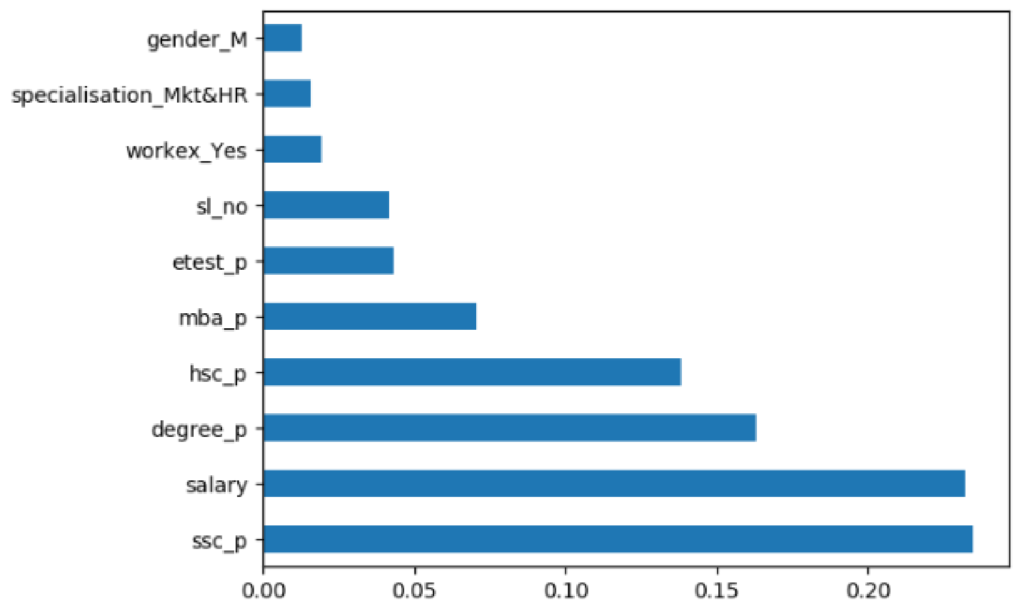
```
# Sort and display the top 10 most important features
top_features = importance.sort_values(ascending=False).head(10)

print("\n--- Top 10 Factors Predicting Placement Status ---")
print(top_features)

# Visualize the top factors
top_features.plot(kind='barh')
plt.title('Feature Importance for Placement Prediction')
plt.show()
```

```
--- Top 10 Factors Predicting Placement Status ---
ssc_p                      0.234817
salary                     0.231976
degree_p                   0.163158
hsc_p                      0.138061
mba_p                      0.070730
etest_p                    0.043434
sl_no                      0.041845
workex_Yes                 0.019243
specialisation_Mkt&HR      0.015943
gender_M                   0.013056
dtype: float64
```

Conclusion from the Model Output

The **top factors** in the output list (e.g., ssc_p, degree_p, hsc_p, are the **primary reasons** driving the placement outcome. ('salary' should not be considered since it is not the reason for placement)

- Students with low scores in 'ssc_p' are more likely to be non-placed.
- The gender has no major influence on the outcome.



4)What kind of relation between salary and mba_p?

```
correlation = dataset['hsc_p'].corr(dataset['salary'])
print("Correlation between hsc_p and salary:", correlation)
```

```
Correlation between hsc_p and salary: -0.04437896596620006
```

5)Which specialization is getting minimum salary?

```
In [24]: #minsalary = Ndataset['salary'].min()
         df = Ndataset

         min_salary_row = df.loc[df['salary'].idxmin()]
         print("Specialization with minimum salary:", min_salary_row['specialisation'])
         print("Minimum salary:", min_salary_row['salary'])
```

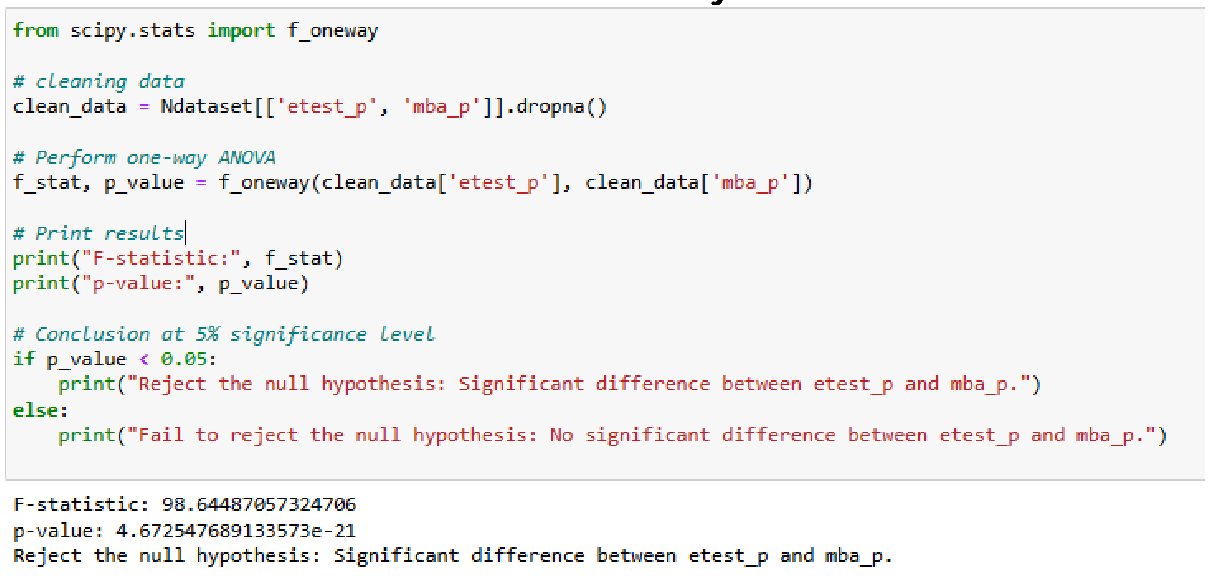
```
Specialization with minimum salary: Mkt&Fin
Minimum salary: 200000.0
```

6)How many of them getting above 500000 salaries?

```
In [10]: salarycount = (Ndataset['salary']>500000).sum()
         salarycount

Out[10]: 3
```

7)Test the Analysis of Variance between etest_p and mba_p at signifance level 5%. (Make decision using Hypothesis Testing)

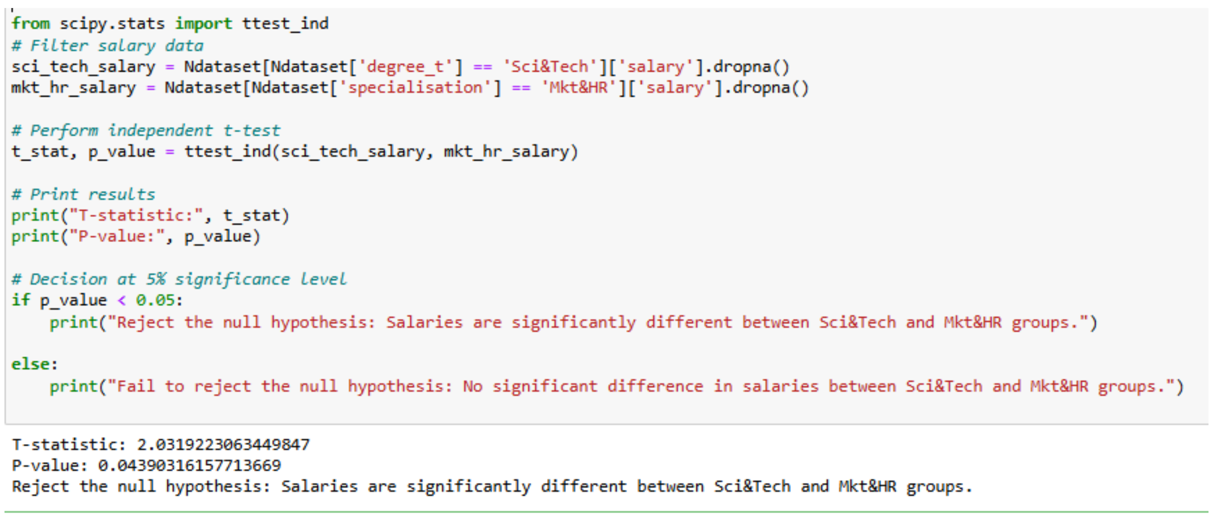
Inferential Analysis



```
from scipy.stats import f_oneway

# cleaning data
clean_data = Ndataset[['etest_p', 'mba_p']].dropna()

# Perform one-way ANOVA
f_stat, p_value = f_oneway(clean_data['etest_p'], clean_data['mba_p'])

# Print results
print("F-statistic:", f_stat)
print("p-value:", p_value)

# Conclusion at 5% significance level
if p_value < 0.05:
    print("Reject the null hypothesis: Significant difference between etest_p and mba_p.")
else:
    print("Fail to reject the null hypothesis: No significant difference between etest_p and mba_p.")
```

```
F-statistic: 98.64487057324706
p-value: 4.672547689133573e-21
Reject the null hypothesis: Significant difference between etest_p and mba_p.
```

8)Test the similarity between the degree_t (Sci&Tech) and specialisation (Mkt&HR) with respect to salary at significance level of 5%.(Make decision using Hypothesis Testing)



```
from scipy.stats import ttest_ind
# Filter salary data
sci_tech_salary = Ndataset[Ndataset['degree_t'] == 'Sci&Tech']['salary'].dropna()
mkt_hr_salary = Ndataset[Ndataset['specialisation'] == 'Mkt&HR']['salary'].dropna()

# Perform independent t-test
t_stat, p_value = ttest_ind(sci_tech_salary, mkt_hr_salary)

# Print results
print("T-statistic:", t_stat)
print("P-value:", p_value)

# Decision at 5% significance level
if p_value < 0.05:
    print("Reject the null hypothesis: Salaries are significantly different between Sci&Tech and Mkt&HR groups.")

else:
    print("Fail to reject the null hypothesis: No significant difference in salaries between Sci&Tech and Mkt&HR groups.")
```

```
T-statistic: 2.0319223063449847
P-value: 0.04390316157713669
Reject the null hypothesis: Salaries are significantly different between Sci&Tech and Mkt&HR groups.
```

9)Convert the normal distribution to standard normal distribution for salary column

Apply z-score normalization:

z-score normalization transforms the data so that it has a mean of 0 and standard deviation of 1.

This is useful for comparing values across different scales or preparing data for machine learning models

Inferential Analysis

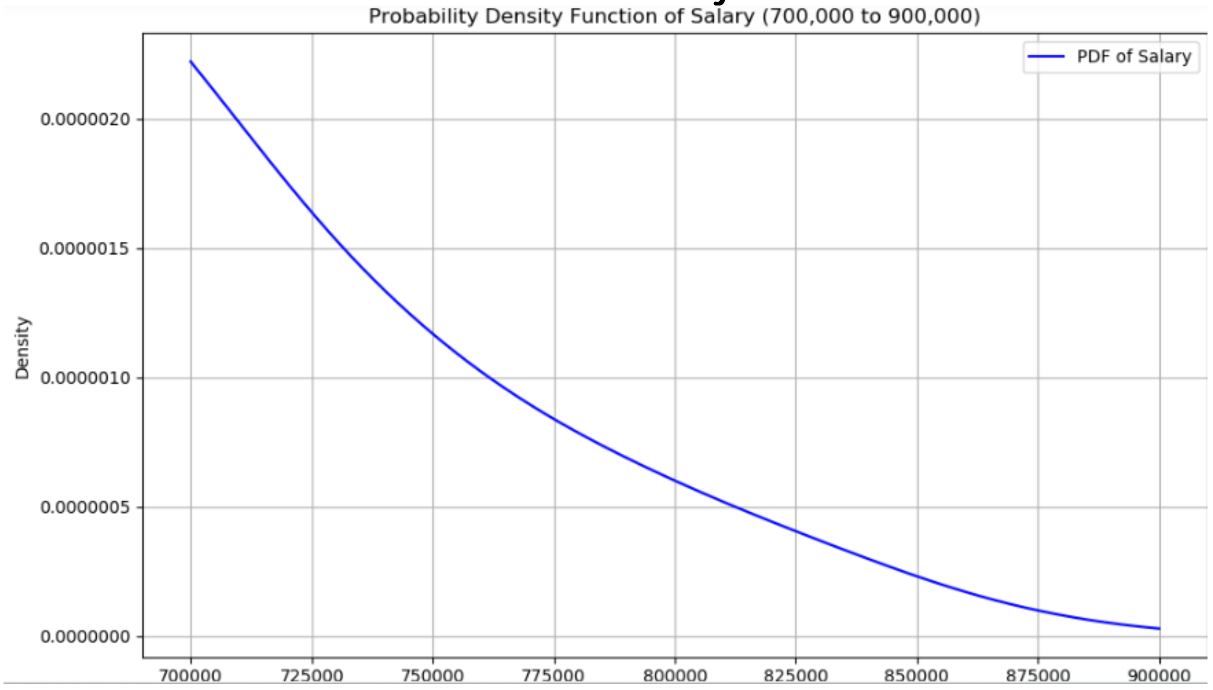
```
from scipy.stats import zscore

Ndataset = Ndataset.dropna(subset=['salary'])

# Apply z-score normalization
Ndataset['salary_zscore'] = zscore(Ndataset['salary'])

# View the result
print(Ndataset[['salary', 'salary_zscore']].head())
```

```
          salary  salary_zscore
0  270000.000000      -0.198966
1  200000.000000      -2.061326
2  250000.000000      -0.731069
3  288655.405405       0.297363
4  346638.513514       1.840012
```

10)What is the probability Density Function of the salary range from 700000 to 900000?

```
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

# Sample salary data (replace with your actual dataset)
np.random.seed(0)
salary_data = np.random.normal(loc=600000, scale=100000, size=1000)

# Convert to DataFrame
df = pd.DataFrame({'salary': salary_data})

# Drop missing values
df = df.dropna(subset=['salary'])

# Define salary range
salary_range = np.linspace(700000, 900000, 500)

# Fit Gaussian KDE
kde = gaussian_kde(df['salary'])

# Evaluate PDF
pdf_values = kde(salary_range)

# Plot PDF
plt.figure(figsize=(10, 6))
plt.plot(salary_range, pdf_values, color='blue', label='PDF of Salary')
plt.title('Probability Density Function of Salary (700,000 to 900,000)')
plt.xlabel('Salary')
plt.ylabel('Density')
plt.grid(True)

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

Inferential Analysis



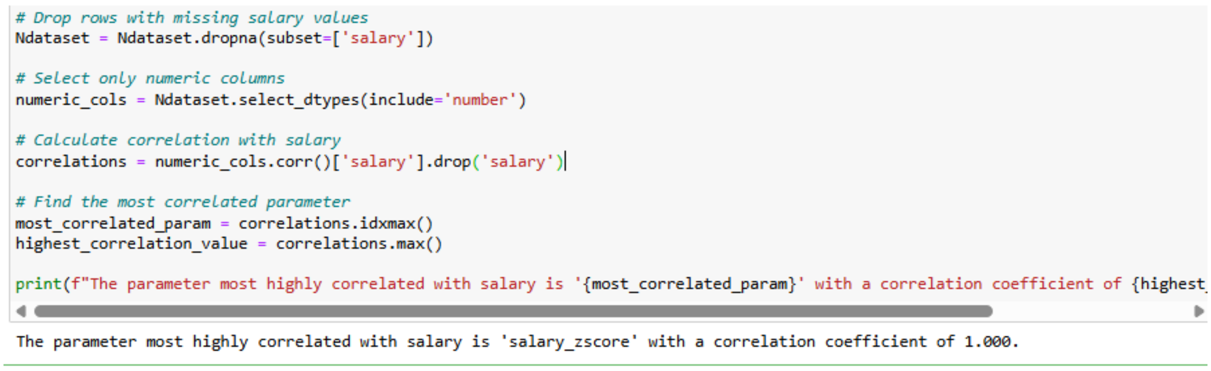
11)Test the similarity between the degree_t(Sci&Tech)with respect to etest_p and mba_p at significance level of 5%.(Make decision using Hypothesis Testing)

```
from scipy.stats import ttest_rel

# Filter and clean data
etest = Ndataset['etest_p']
mba = Ndataset['mba_p']

# Drop missing values and align indices
paired_data = Ndataset[['etest_p', 'mba_p']].dropna()
etest_clean = paired_data['etest_p']
mba_clean = paired_data['mba_p']

# Perform paired t-test
t_stat, p_value = ttest_rel(etest_clean, mba_clean)

print("t-statistic:", t_stat)
print("p-value:", p_value)

# Conclusion at 5% significance level
if p_value < 0.05:
    print("Reject the null hypothesis: Significant difference between etest_p and mba_p.")
else:
    print("Fail to reject the null hypothesis: No significant difference between etest_p and mba_p.")
```

```
t-statistic: 10.840680769176654
p-value: 4.1449406968815296e-22
Reject the null hypothesis: Significant difference between etest_p and mba_p.
```

Inferential Analysis

12)Which parameter is highly correlated with salary?



```
# Drop rows with missing salary values
Ndataset = Ndataset.dropna(subset=['salary'])

# Select only numeric columns
numeric_cols = Ndataset.select_dtypes(include='number')

# Calculate correlation with salary
correlations = numeric_cols.corr()['salary'].drop('salary')

# Find the most correlated parameter
most_correlated_param = correlations.idxmax()
highest_correlation_value = correlations.max()

print(f"The parameter most highly correlated with salary is '{most_correlated_param}' with a correlation coefficient of {highest_
```

```
The parameter most highly correlated with salary is 'salary_zscore' with a correlation coefficient of 1.000.
```

---

13) Plot any useful graph and explain it.

A **Factorial Interaction Plot** (also known as a **Means Plot**) is the most useful graph to visualize the results of a Two-Way ANOVA. It directly illustrates the **main effects** and, more importantly, the **interaction effect**.

Factorial Interaction Plot

Why It's Useful

1. **Interaction Check:** It clearly shows if the lines are parallel. **Non-parallel lines** suggest a significant interaction, which the ANOVA table tests.
2. **Main Effect Visualization:** It shows the overall mean differences for each factor.
3. **Data Interpretation:** It makes the nature of non-significant effects visually intuitive. For example, our plot will likely show that the mean lines are close to parallel and close together.

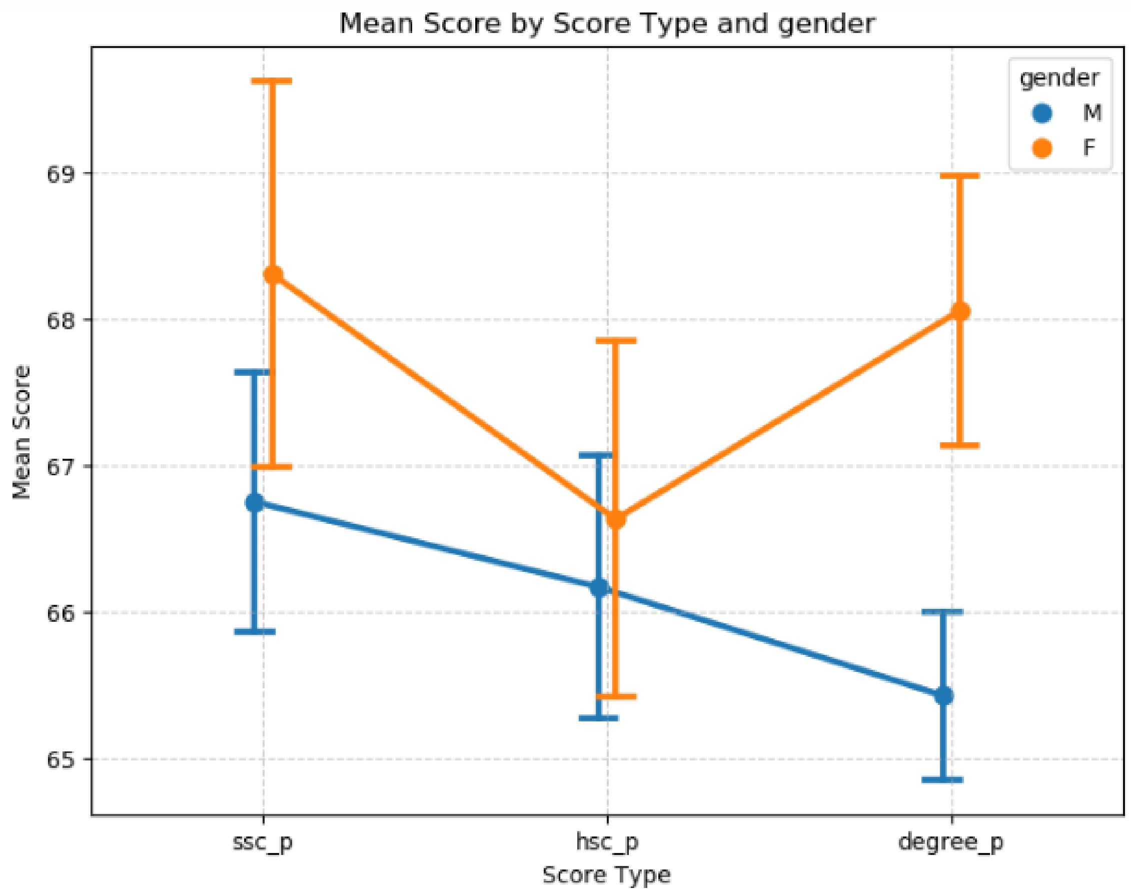
The second factor is 'gender' and the melted columns are 'Score_Type' ['ssc_p', 'hsc_p', 'degree_p') and 'Score':

Python

Inferential Analysis

```python
# The second factor (your C(gender) column)
FACTOR_2 = 'gender'

data_melted = pd.melt(
    dataset,
    id_vars=[FACTOR_2],
    value_vars=['ssc_p', 'hsc_p', 'degree_p'],
    var_name='Score_Type',
    value_name='Score'
)

# Create the plot
plt.figure(figsize=(8, 6))

# Use the 'pointplot' to show means and confidence intervals
sns.pointplot(
    data=data_melted,
    x='Score_Type',         # Factor 1 on the X-axis (SSC, HSC, Degree)
    y='Score',              # Dependent Variable on the Y-axis
    hue=FACTOR_2,           # Factor 2 (Gender) as separate lines/colors
    capsize=0.1,            # Adds caps to the error bars (CI)
    dodge=True,             # Ensures points for the same score type are separated
    errorbar='se'           # Show standard error (or 'ci' for confidence interval)
)

plt.title(f'Mean Score by Score Type and {FACTOR_2}')
plt.xlabel('Score Type')
plt.ylabel('Mean Score')
plt.legend(title=FACTOR_2)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Inferential Analysis



Interpretation from the Plot

Factorial Interaction Plot (Means Plot) from our Two-Way ANOVA, visualizing the effect of **Score Type** (ssc_p, hsc_p, degree_p) and **Gender** (M, F) on the **Mean Score**.

The interpretation confirms the statistical conclusions we reached earlier (where all effects were non-significant at $\alpha=0.05$).

Interpretation of the Plot

1. Interaction Effect (Lines' Parallelism)

The key to interpreting interaction is looking at the lines' trends:

- **Observation:** The two lines (Male in blue, Female in orange) are **not parallel**. They are closest together at hsc_p and furthest apart at ssc_p.
- **Conclusion:** The lines crossing or converging/diverging suggests a potential interaction effect—meaning the difference between male and female scores **depends on the Score Type**.
- **ANOVA Confirmation:** our ANOVA result for the interaction term ($P \approx 0.55$) indicated this visual difference was **not statistically significant**. Therefore, while

there's a visible trend, we conclude the interaction effect is not strong enough to be considered reliable in the population.

2. Main Effect of Gender (Vertical Separation)

This effect is determined by the overall vertical distance between the two lines:

- **Observation:** The **orange line (Female)** is consistently **above** the **blue line (Male)** across all three score types.
- **Conclusion:** This suggests that the **Mean Score for Females is generally higher** than the Mean Score for Males.
- **ANOVA Confirmation:** Our ANOVA result for gender was $P \approx 0.053$. This confirms the visual observation is a **borderline case**. While visually apparent, the effect is **not statistically significant** at the standard $\alpha=0.05$ level.

3. Main Effect of Score Type (Overall Trend Across X-Axis)

This effect is determined by the general trend of all points (ignoring gender):

- **Observation:** The mean scores are roughly in the range of 66 to 68.5. There is a slight **dip at hsc_p** for both genders and then a rise.
- **Conclusion:** This suggests that the Score Type has a minimal overall impact on the score average.
- **ANOVA Confirmation:** Our ANOVA result for Score Type was $P \approx 0.51$. This confirms the visual observation that the differences between the overall average score for ssc_p, hsc_p, and degree_p are **not statistically significant**.

Summary

The plot visually represents the small, non-significant differences found in our ANOVA:

- Females generally score higher, but the difference isn't definitively significant.
- The relative performance between genders changes slightly depending on the score type (e.g., the gender difference is smaller in hsc_p), but this change is not statistically significant.
- The type of score itself has no significant overall impact on the average score.
