How to remove multicollinearity for a regression model

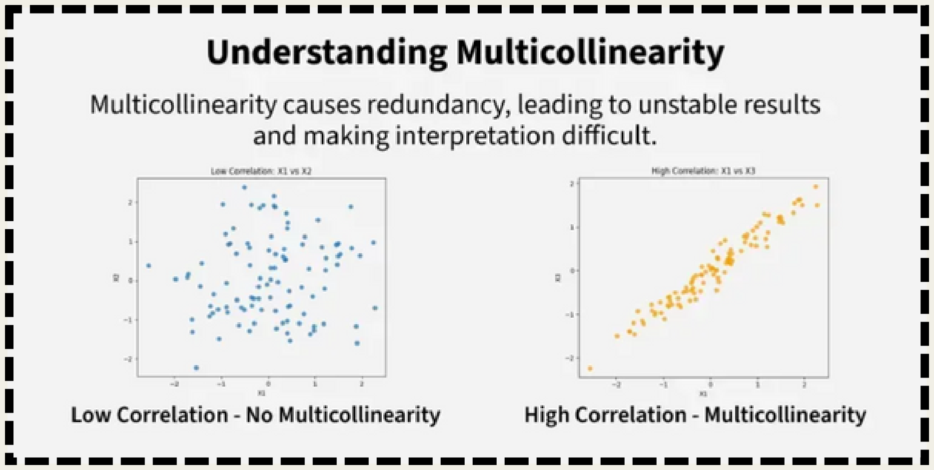
Before you create a regression model, check on the multicollinearity of your predictor variables. It might save you a lot of trouble!

Multicollinearity

- Multicollinearity happens when two or more predictor(independent) variables in a model are closely related to each other
- Because they give similar information, it becomes difficult to know how each one affects the result.
- This is a common problem in **multiple linear regression** and can make the model's results less reliable

Problems with Multicollinearity

- **Unstable Coefficients:** When independent variables are highly correlated, small changes in the data can cause large fluctuations in the regression coefficients
- **Reduced Interpretability:** Since correlated variables provide similar information, it's challenging to find the individual contribution of each predictor.
- **Risk of Overfitting**: The model may fit the training data too closely by capturing random noise which decrease its ability to work well on new data.



Detecting Multicollinearity

1. **Variance Inflation Factor:**

- It measures how much the variance of an estimated ***regression coefficient*** is increased due to the correlation among the predictors.
- ***VIF*** is a common and effective method for detecting multicollinearity.

VIF = 1	→	no correlation
1 < VIF < 5	→	moderate correlation
VIF > 5	→	high correlation

Code snippet

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
def calc_vif(X):
    vif=pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values,i) for i in range(X.shape[1])]
    return(vif)
```

```
calc_vif(dataset[quan])
```

	variables	VIF
0	ssc_p	67.026700
1	hsc_p	56.131492
2	degree_p	112.755275
3	etest_p	33.696391
4	mba_p	108.585463
5	salary	15.167704

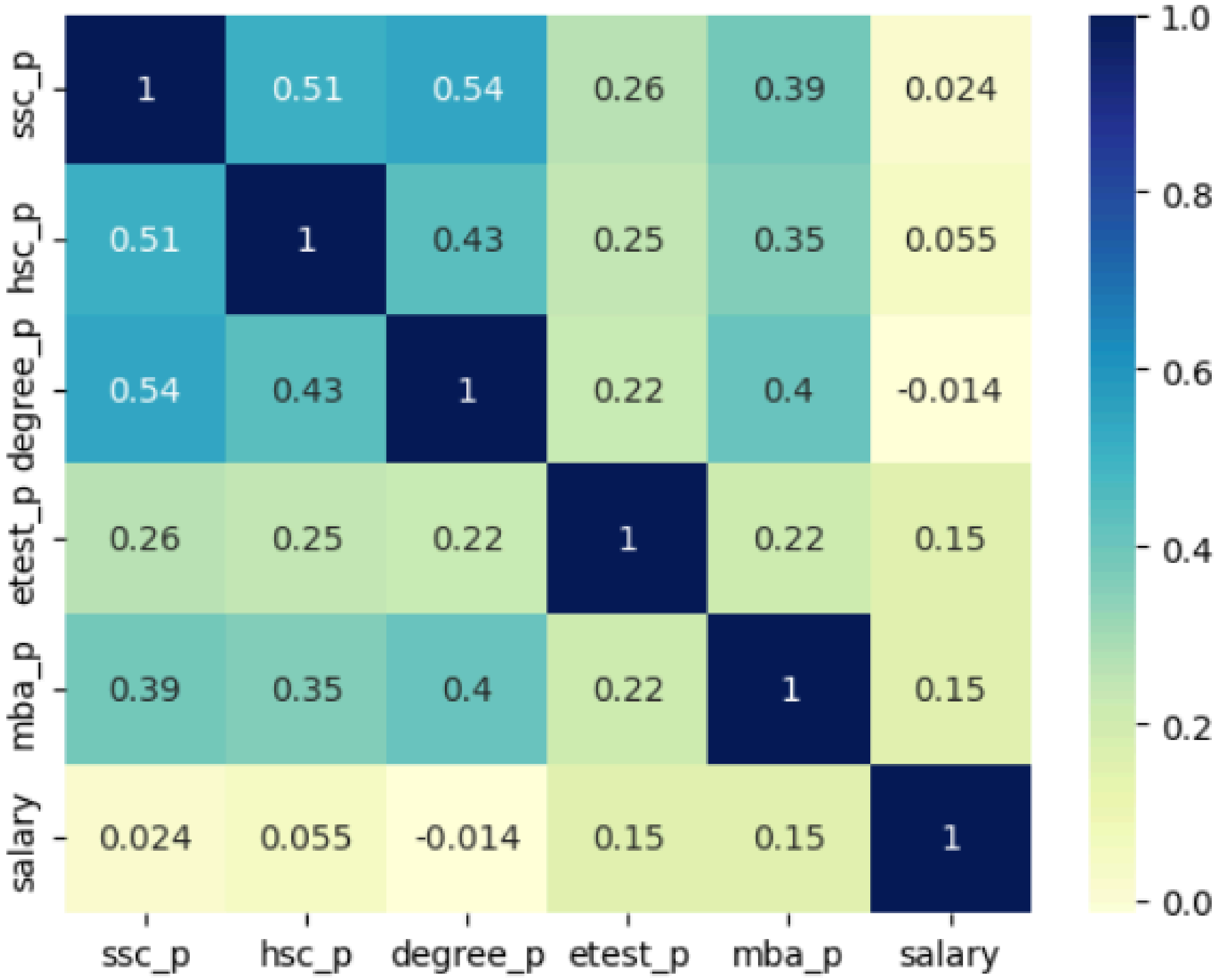
Detecting Multicollinearity

2. **Correlation Matrix:**
   - A correlation matrix can visually show the relationships between all pairs of independent variables.
   - If the correlation between two variables is high (***typically above 0.8***), it may indicate the presence of multicollinearity.
   - *However, this method can miss multicollinearity involving three or more variables.*

Code snippet

```python
import matplotlib.pyplot as plt
import seaborn as sns
corr_matrix=dataset.corr(numeric_only=True)
sns.heatmap(corr_matrix, cmap="YlGnBu", annot=True)
plt.show
```

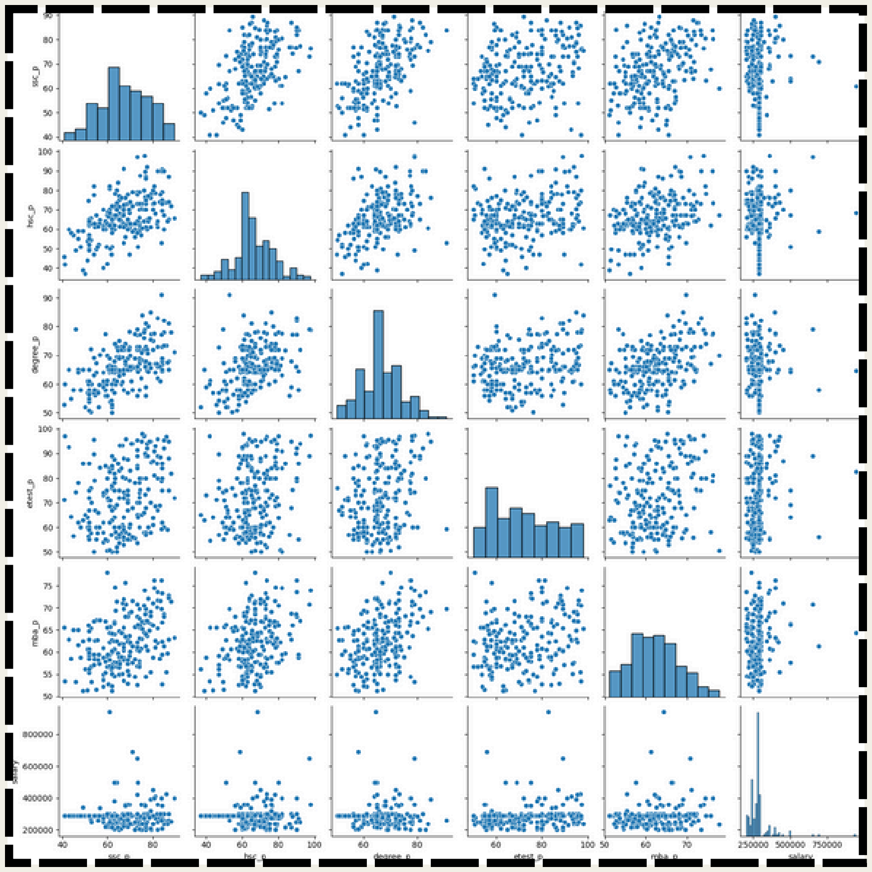
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Detecting Multicollinearity

3.Scatterplot Matrix:

- A scatterplot matrix provides a visual representation of the relationships between all pairs of independent variables.
- If one of the scatterplots shows a strong linear relationship, it could indicate multicollinearity.



Solutions To Multicollinearity In Linear Regression

Remove Highly Correlated Predictors

- If two or more variables are highly correlated, consider removing one of them

Implementation example:

- Consider our dataset has ssc_p, hsc_p, degree_p,mba_p, ent_p as predictor variables for salary estimation

```
calc_vif(dataset[quan])
```

	variables	VIF
0	ssc_p	67.026700
1	hsc_p	56.131492
2	degree_p	112.755275
3	etest_p	33.696391
4	mba_p	108.585463
5	salary	15.167704

- Here, all the variables have high VIF (***value > 5***), so we removed all of them except ent_p and salary
- However, the VIF value for the etest_p and salary is not reduced after removing other variables, so we need to try other solutions to remove multicolinearity for our dataset

```
calc_vif(dataset[["etest_p","salary"]])
```

	variables	VIF
0	etest_p	11.944567
1	salary	11.944567

Principal Component Analysis:

- PCA reduces the dimensionality of the dataset by combining correlated predictors into a smaller number of uncorrelated components.
- This can retain most of the explanatory power while eliminating multicollinearity

Implementation Example:

- Here we are combining all the mark columns into a single component using PCA

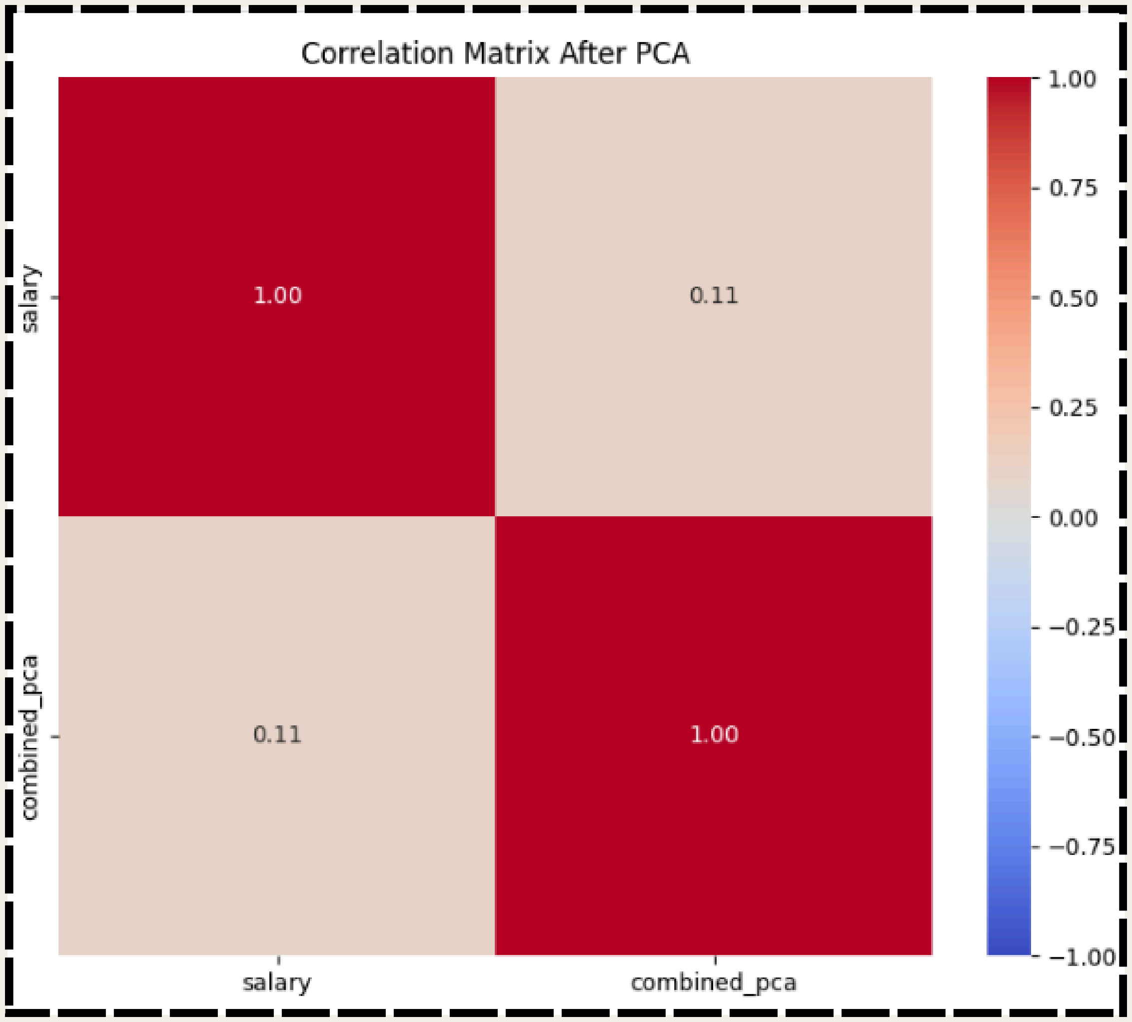
```
from sklearn.decomposition import PCA
# Apply PCA to the highly correlated features: hsc_p,ssc_p,degree_p,mba_p

pca = PCA(n_components=1)  # Combine them into 1 component
X_combined = dataset[['ssc_p', 'hsc_p', 'degree_p','mba_p','etest_p']]
X_pca = pca.fit_transform(X_combined)
# Add the PCA result as a new feature to the dataset
dataset['combined_pca'] = X_pca
# Now, we will drop the original features and use the PCA component
dataset_reduced = dataset.drop(columns=['ssc_p', 'hsc_p', 'degree_p','mba_p','etest_p'])
corr_after_pca = dataset_reduced.corr(numeric_only=True)
# Calculate VIF after PCA (for the reduced set)
print("VIF after pca")
calc_vif(dataset_reduced[["combined_pca","salary"]])
```

```
VIF after pca
```

	variables	VIF
0	combined_pca	1.00085
1	salary	1.00085

- After implementing PCA, the ***VIF*** values of the **combined_PCA** and the **salary** were reduced to a moderate correlation
- After implementing PCA, the Correlation matrix of the dataset is as shown below



Other solutions

Below are some of the other solutions used to remove multicollinearity from the dataset

- Use Ridge(L1 Regularization) or Lasso regression(L2 Regularization)
- Increasing the sample size can reduce the variance of the coefficient estimates and improve the reliability of the regression model.
- Apply standardization, which can help reduce multicollinearity in some cases.

Thank you

Like

& FOLLOW FOR MORE