FUNDAMENTALS OF DATA SCIENCE

MA - 515

Project Report

SUBMITTED TO

Dr. Arun Kumar

PRESENTED BY:

Ponugupati Hemanth	2020EEB1191
Ch. Sai Sharan	2020MMB1342
P. Chandra Shekhar Varma	2020CEB1021

- # Problem Statement:

1. To do EDA on the data 1(Advertising data). Using multiple linear regression, ridge and lasso techniques to predict the Sales. Compare different methods.
2. To do EDA on the data 2(college data). Using multiple linear regression, ridge and lasso techniques to predict the graduation rate . Compare different methods.
3. To apply PCA on data 3(HD Data) to reduce it to 10 columns and apply EDA on the reduced data. Now use linear regression, LASSO and RIDGE regression on data 3 and compare the methods.
4. Use SVD on the data set 3(HD Data) and reduce it to 50% of the original data.

- # Data description:

1) Advertising dataset was allocated to me. It consists of manufacturing data of TV, Radio, newspaper and their sales. The task is to predict 5the sales of the given components using the methods mentioned above.
2) The dataset is of the shape (201, 5).
3) College dataset was allocated to me. It consists of data of students studying in the university(containing different features) and their graduation rate. The task is to predict the graduation rate using those features using the methods mentioned above.
4) The dataset is of the shape (778,18). The shape indicates that we have 17 features, 1 target column and 505 data points
5) HD Data dataset was allocated to me. It consists of features Y and X1 through X150. The task is to analyze and predict the quantity Y of the dataset using various regression techniques.
6) The dataset has the shape of ().The shape indicates that we have __ features and __ target column and __ data points.

- # Problem Statement 1:

    - **Exploratory Data Analysis:**

2. I have described the data using df.describe() and got their data types using df.dtypes.

```
df.describe()
```
✓ 0.0s

	Unnamed: 0	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

```
df.dtypes
```

0.0s

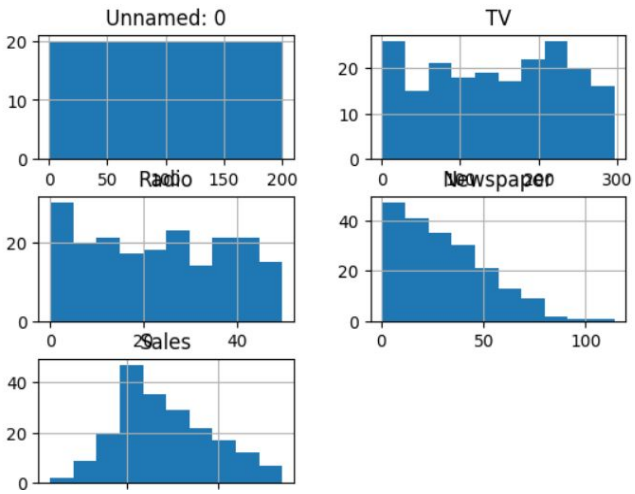
```
Unnamed: 0       int64
TV             float64
Radio          float64
Newspaper      float64
Sales          float64
dtype: object
```

2)Then we plotted the histograms for each of the features using df.hist().

```
df.hist()
```

[19] 0.4s

```
array([[<Axes: title={'center': 'Unnamed: 0'}>,
        <Axes: title={'center': 'TV'}>],
       [<Axes: title={'center': 'Radio'}>,
        <Axes: title={'center': 'Newspaper'}>],
       [<Axes: title={'center': 'Sales'}>, <Axes: >]], dtype=object)
```




```
df.corr()
```

[20] 0.0s

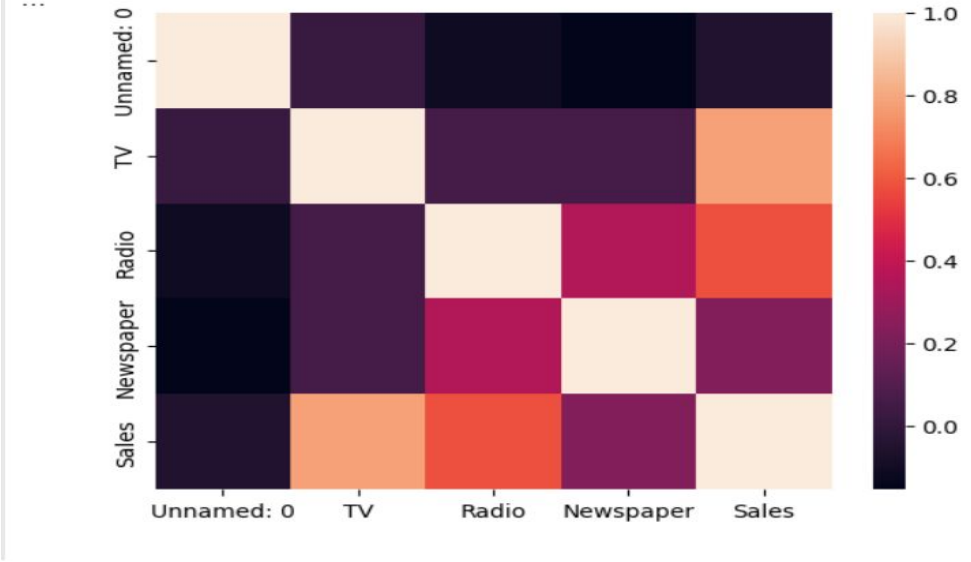
	Unnamed: 0	TV	Radio	Newspaper	Sales
Unnamed: 0	1.000000	0.017715	-0.110680	-0.154944	-0.051616
TV	0.017715	1.000000	0.054809	0.056648	0.782224
Radio	-0.110680	0.054809	1.000000	0.354104	0.576223
Newspaper	-0.154944	0.056648	0.354104	1.000000	0.228299
Sales	-0.051616	0.782224	0.576223	0.228299	1.000000

4)plotted the correlating graph and the heat map for the correlating values of the features given.

```
sns.heatmap(df.corr())
```

[21] ✓ 0.1s

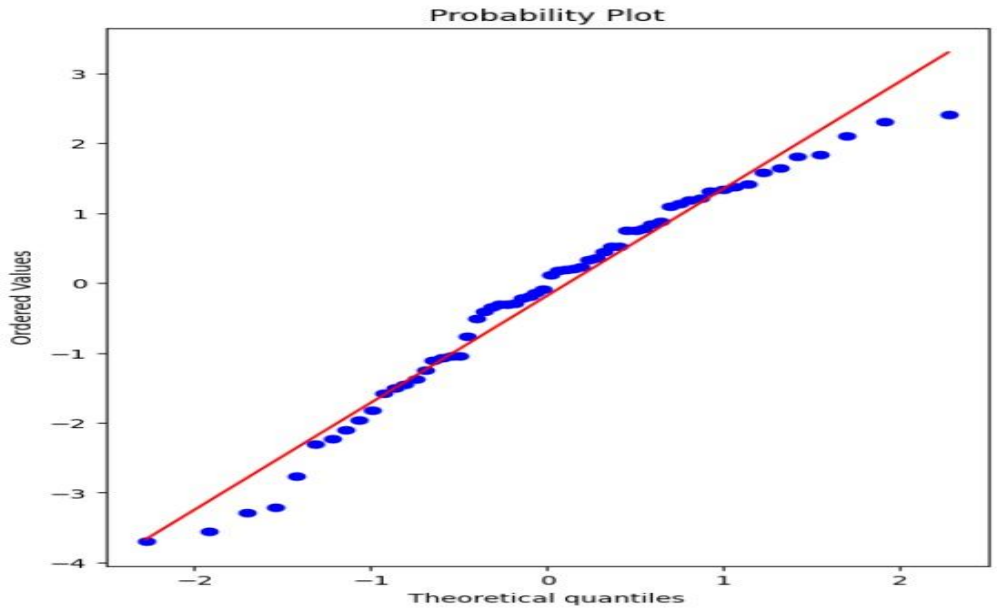
<Axes: >



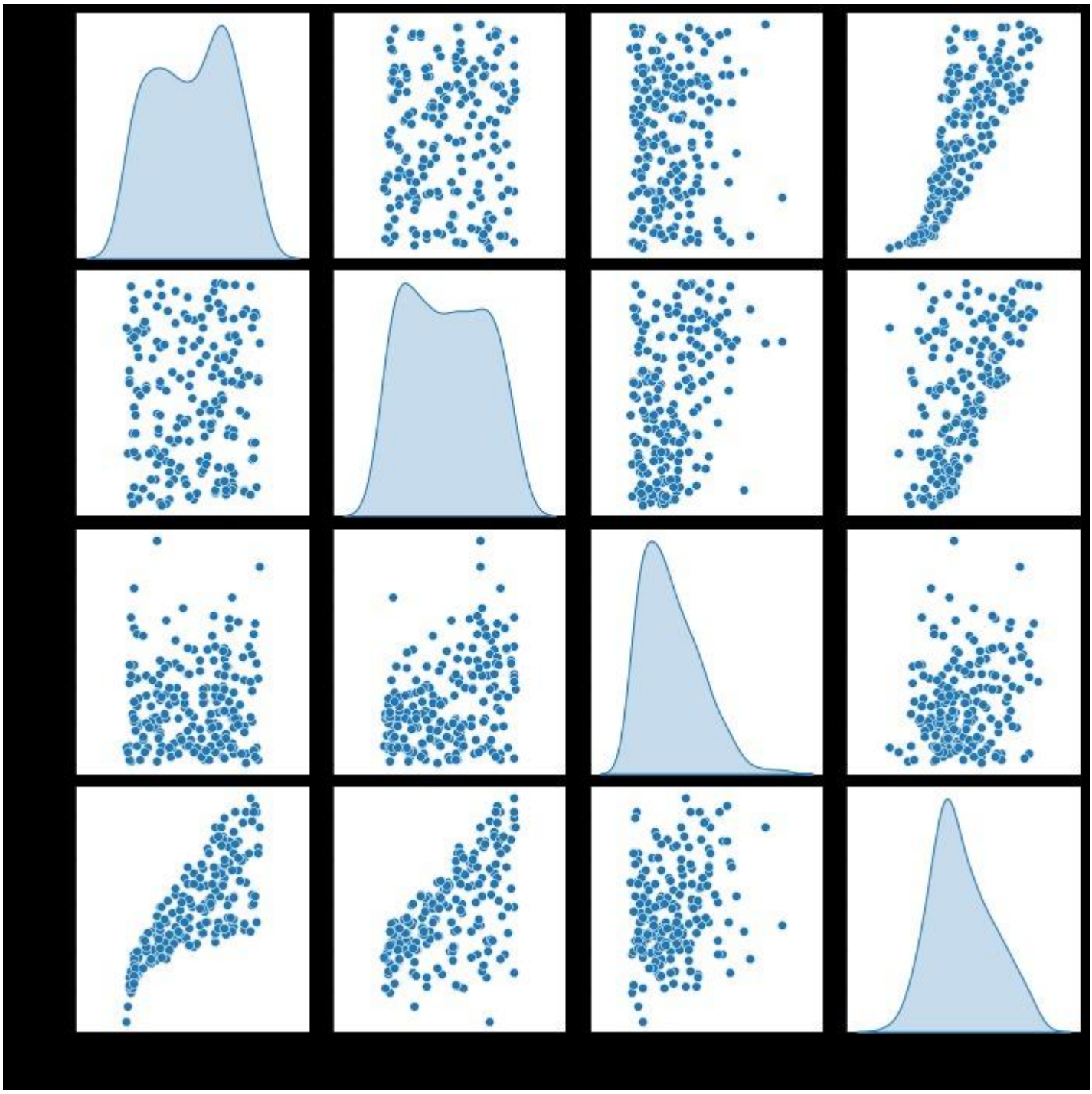
.

PP plot:

- pp plot gives an estimation of how good our data is compared to theoretical quantities



Pair plot:



0 50 100 150 200 250 300 0

```
]: # Relationships between features
   sns.pairplot(df,diag_kind='kde')
```

```
]: <seaborn.axisgrid.PairGrid at 0x216014fb648>
```

- Model development train test split:

1)Multi linear regression:
Train test split: 70:30
RMSE: 0.664643175

2)Ridge regression:
Train test split: 70:30
RMSE: 0.618071993

2)Lasso regression:
Train test split: 70:30
RMSE: 1.13080010

- **RESULTS:**

Comparison with multi linear , ridge , lasso we got our best fit through ridge regression as it has minimum RMSE.

- **Conclusion:**

In dataset - 1 , we can conclude that investment on TV gives better returns / sales compared to radio or newspaper.

■ EDA on College.csv

2. data.head()

1) I have described the data using data.head(), found out the related information like data types of different features using data.info().

2)Then we plotted the histograms for the first three quantities using “sns.pairplot(data, vars = data.columns[1:4])” and also plotted the heat map of the correlating values for the features given.

```
data.head()
```
[5] ✓ 0.0s Python

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio	perc.alumni	Ex
0	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18.1	12	
1	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	12.2	16	1
2	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	12.9	30	
3	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7.7	37	1
4	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11.9	2	1

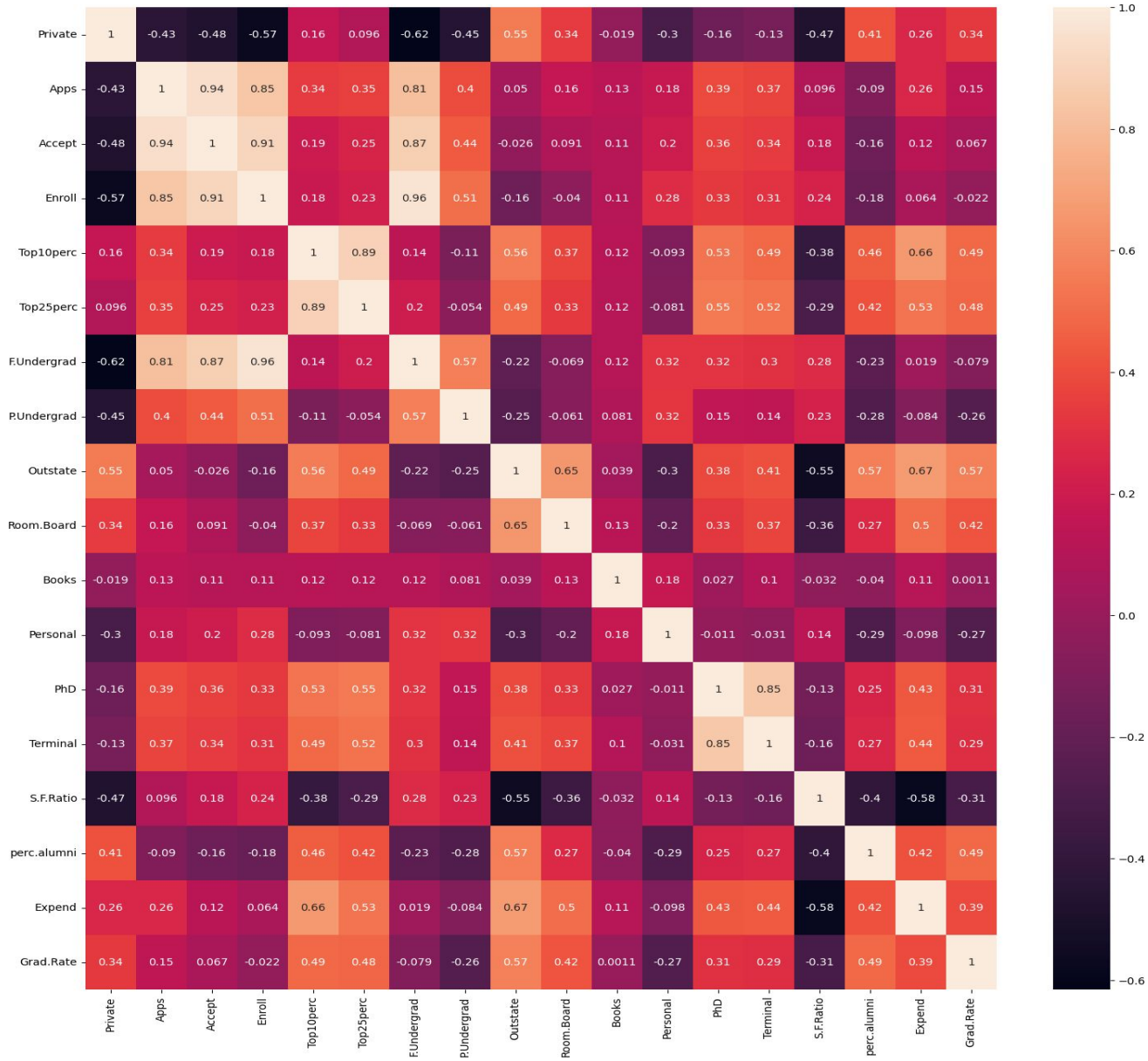
```
data.info()
```
[6] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 18 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Private       777 non-null    object 
 1   Apps          777 non-null    int64  
 2   Accept        777 non-null    int64  
 3   Enroll        777 non-null    int64  
 4   Top10perc     777 non-null    int64  
 5   Top25perc     777 non-null    int64  
 6   F.Undergrad   777 non-null    int64  
 7   P.Undergrad   777 non-null    int64  
 8   Outstate      777 non-null    int64  
 9   Room.Board    777 non-null    int64  
 10  Books         777 non-null    int64  
 11  Personal      777 non-null    int64  
 12  PhD           777 non-null    int64  
 13  Terminal      777 non-null    int64  
 14  S.F.Ratio     777 non-null    float64
 15  perc.alumni   777 non-null    int64  
 16  Expend        777 non-null    int64  
 17  Grad.Rate     777 non-null    int64  
dtypes: float64(1), int64(16), object(1)
memory usage: 109.4+ KB
```

```
data.describe()
```
✓ 0.0s Python

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio	perc.alu
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000
mean	0.727156	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	855.298584	10440.669241	4357.526384	549.380952	1340.642214	72.660232	79.702703	14.089704	22.743
std	0.445708	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	1522.431887	4023.016484	1096.696416	165.105360	677.071454	16.328155	14.722359	3.958349	12.391
min	0.000000	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	1.000000	2340.000000	1780.000000	96.000000	250.000000	8.000000	24.000000	2.500000	0.000
25%	0.000000	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	95.000000	7320.000000	3597.000000	470.000000	850.000000	62.000000	71.000000	11.500000	13.000
50%	1.000000	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	353.000000	9990.000000	4200.000000	500.000000	1200.000000	75.000000	82.000000	13.600000	21.000
75%	1.000000	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	967.000000	12925.000000	5050.000000	600.000000	1700.000000	85.000000	92.000000	16.500000	31.000
max	1.000000	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000	21836.000000	21700.000000	8124.000000	2340.000000	6800.000000	103.000000	100.000000	39.800000	64.000

Heat map for the college data




```
plt.figure(figsize = (18,18))
sns.heatmap(data.corr(), annot=True)
```

✓ 1.8s

```
<Axes: >
```

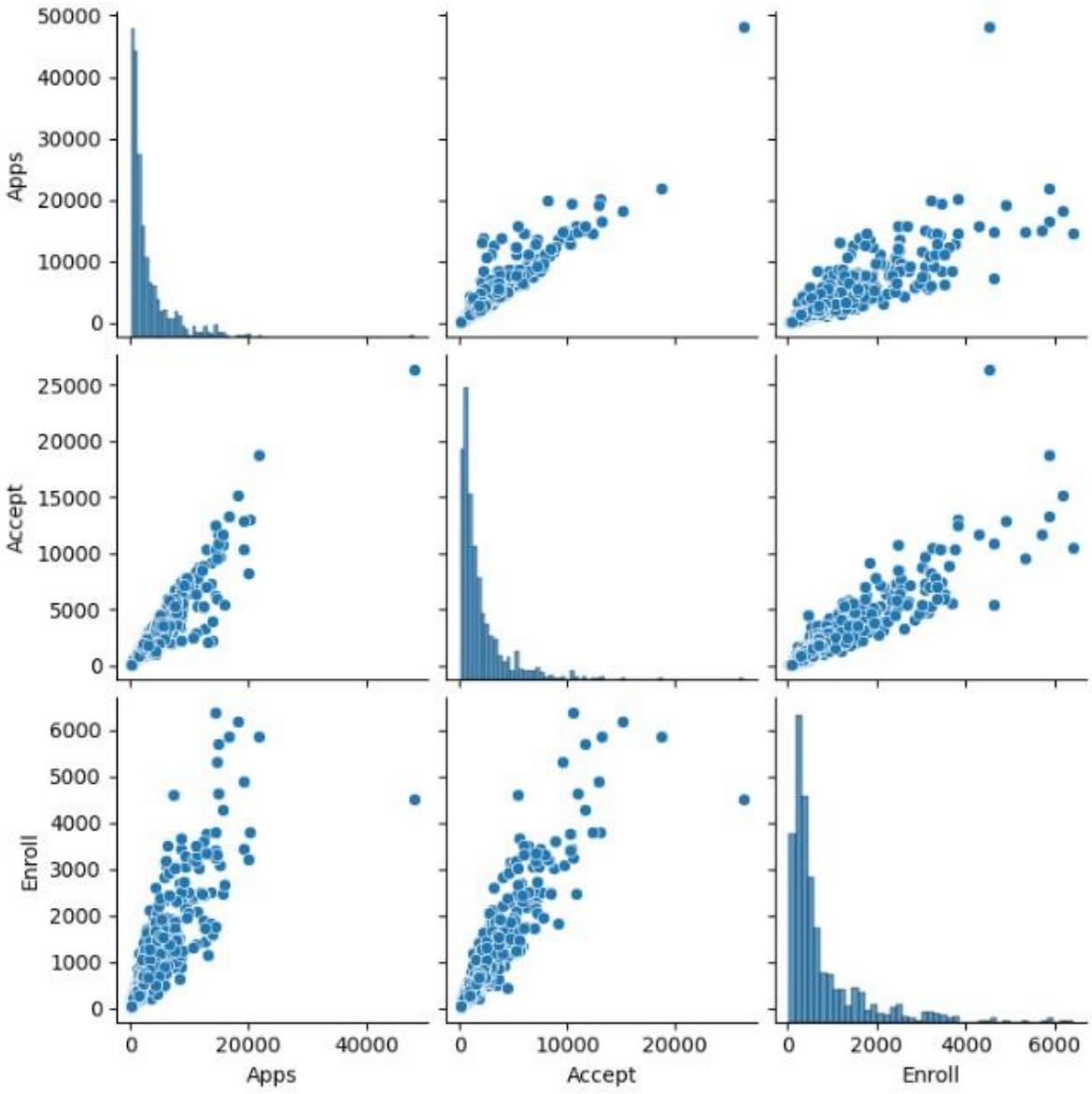
3. We can observe from the above heatmap that the features Apps, Accept and Enroll have high correlation between any two chosen pairs out of those 3 features. Since our aim is to find the number of accepted applications, we can say that the number of applications received and number of enrollments will highly influence the number of accepted applications. It can be verified by plotting a pairplot as shown in figure.

Pairplot of the college data

```
sns.pairplot(data, vars=data.columns[1:4])
```

[11] ✓ 3.0s

```
<seaborn.axisgrid.PairGrid at 0x13f863b32e0>
```

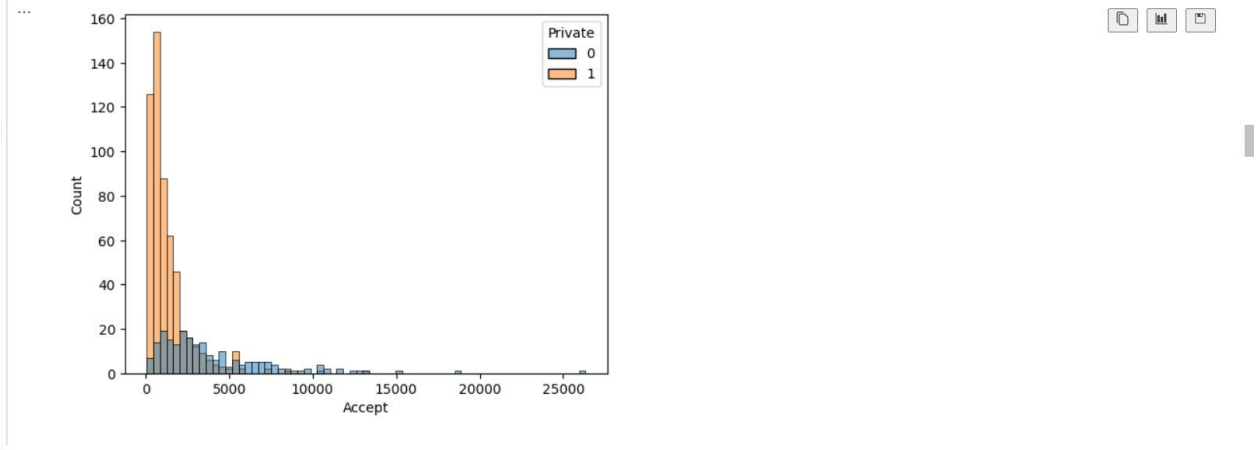


4)We can also infer from the heatmap that the number of accepted applications and the number of full time undergraduates have a decent correlation. Their corresponding pairplot is shown in the above figure.

```
sns.histplot(data,x='Accept', hue='Private')
```

[14] ✓ 0.5s Python

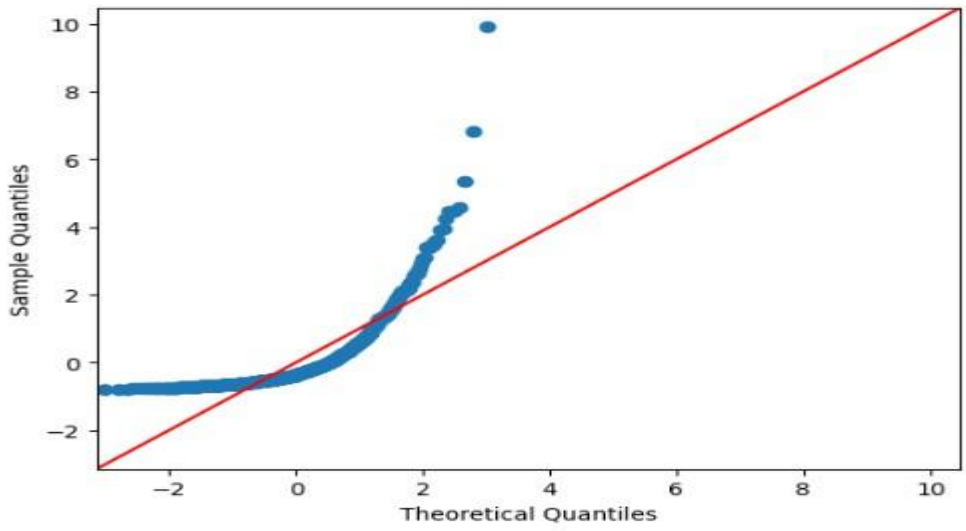
```
<Axes: xlabel='Accept', ylabel='Count'>
```



- Most of the colleges accept less than 5000 applications. It can be verified from the histogram given figure

```
sm.qqplot(data['Accept'], fit=True, line='45')
py.show()
```

[16] ✓ 0.1s



QQ Plot:

- We can also observe that the feature Accept doesn't exactly follow normal distribution. It can be verified from the qq plot as shown in figure.

- **Data Preparation for training and testing:**

1. All the features except Accept are taken as input data. Accept is the output data. The data is being split for training and testing in the ratio of 80:20

- **Ridge Regression:**

1. On applying Ridge Regression for alpha = 1, the score obtained is 0.9580184334821796. On applying Ridge Regression for different values of alpha, the score tended to decrease with increase in alpha and thus the score was maximum at alpha = 0(linear regression).
2. However, it has to be noted that the score peaked at different alpha values for different train_test_split and thus we can conclusively say that max. score is obtained at a particular value of alpha.
3. This is due to the random nature of dataset splitting as it can lead to different data distribution in different splittings. Moreover, it was generally observed that multilinear regression and ridge had similar performance on this dataset.

- **Multilinear Regression:**

1. It is used to estimate the relationship between two or more independent variables and one dependent variable. You can use multiple linear regression when you want to know:
    1. How strong the relationship is between two or more independent variables and one dependent variable.
    2. The value of the dependent variable at a certain value of the independent variables.

Now, multilinear regression is performed on the given training data. The score obtained is 0.9580139427096107. On applying standardization and normalization we obtain scores as 0.9491379181342835 and 0.6829584218103979. Thus, feature scaling proved to be ineffective for the given data.

- **Conclusion:**

Thus, we performed exploratory data analysis on the given data and performed multilinear regression and ridge regression in addition to feature scaling. Both feature scaling methods, that is standardization and normalization, proved to be ineffective for the given data. Multilinear regression and ridge regression had similar performance levels. It was observed that the number of applications accepted relied heavily on the number of applications received and number of enrolments.

HD Data

PCA on the data:

1. PCA algorithm:

- Principal Component Analysis (PCA) reduces data dimensionality by identifying key features through eigenvector decomposition of the covariance matrix, enabling efficient representation while preserving data variance for analysis or modeling.

2. EDA on data after PCA:

```
df_comp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   pc1     1000 non-null   float64
 1   pc2     1000 non-null   float64
 2   pc3     1000 non-null   float64
 3   pc4     1000 non-null   float64
 4   pc5     1000 non-null   float64
 5   pc6     1000 non-null   float64
 6   pc7     1000 non-null   float64
 7   pc8     1000 non-null   float64
 8   pc9     1000 non-null   float64
 9   pc10    1000 non-null   float64
 10  y       1000 non-null   float64
dtypes: float64(11)
memory usage: 86.1 KB
```

```
df_comp.describe()
```

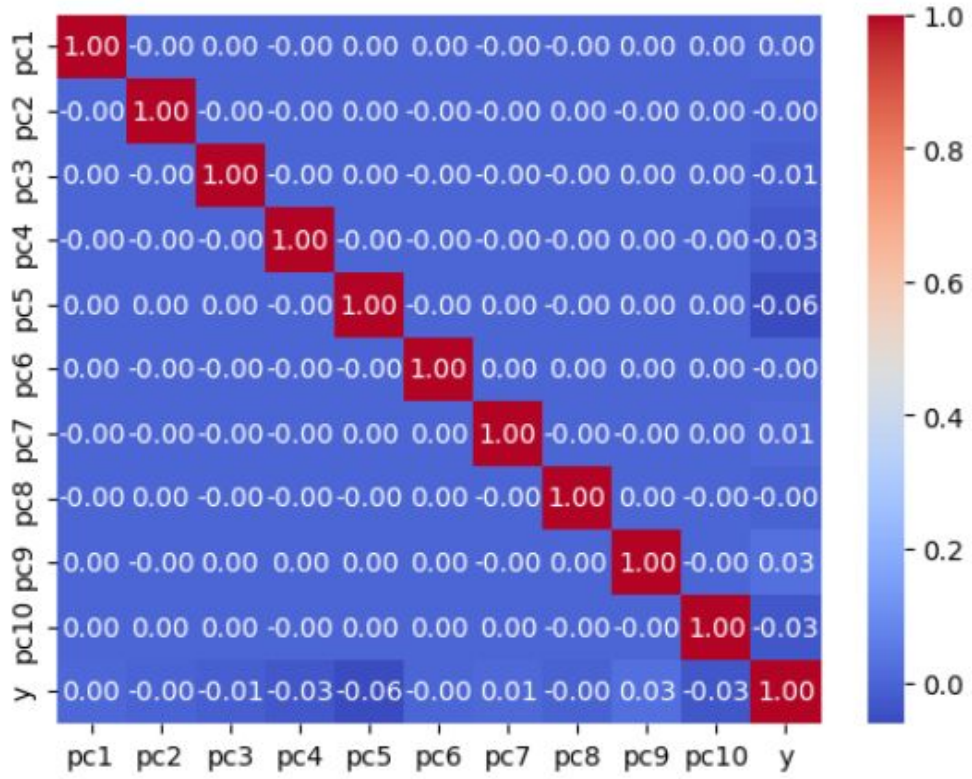
	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8	pc9	pc10	y
count	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03	1000.000000
mean	-3.183231e-15	-1.136868e-16	-1.534772e-15	-3.524292e-15	4.092726e-15	7.901235e-15	-5.115908e-15	5.798029e-15	5.911716e-15	7.730705e-15	274.658472
std	4.232100e+01	4.202718e+01	4.178983e+01	4.110566e+01	4.107151e+01	4.070872e+01	4.010042e+01	3.971851e+01	3.960971e+01	3.921636e+01	217.150211
min	-1.328501e+02	-1.326098e+02	-1.181623e+02	-1.172650e+02	-1.239702e+02	-1.399055e+02	-1.122062e+02	-1.287366e+02	-1.120991e+02	-1.172324e+02	1.102006
25%	-2.802577e+01	-2.839521e+01	-2.943659e+01	-2.834175e+01	-2.752397e+01	-2.765683e+01	-2.744257e+01	-2.365802e+01	-2.653130e+01	-2.505064e+01	101.450158
50%	-1.368522e+00	-1.179830e+00	6.053752e-02	4.128297e-01	-1.180728e+00	2.270592e-01	-4.657890e-01	-2.591222e-01	-1.453056e+00	-9.527041e-01	211.048009
75%	2.628971e+01	2.840360e+01	2.884648e+01	2.855975e+01	2.702217e+01	2.630422e+01	2.757229e+01	2.626931e+01	2.815891e+01	2.694653e+01	398.146774
max	1.359199e+02	1.406453e+02	1.416508e+02	1.521727e+02	1.632387e+02	1.428951e+02	1.562355e+02	1.401481e+02	1.556420e+02	1.185586e+02	993.348132

```
df_comp.duplicated().sum()
```

```
0
```

```
sns.heatmap(df_comp.corr(), annot=True, cmap='coolwarm', fmt='.2f')
```

```
<Axes: >
```



Heat map of principal components

```
df_comp.head()
```

0.0s

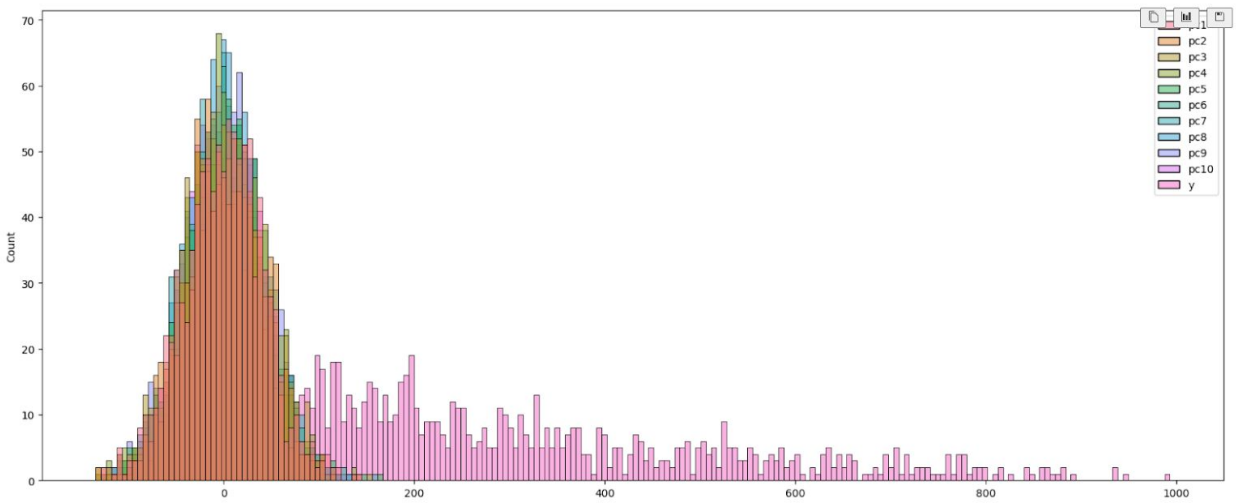
	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8	pc9	pc10	y
0	11.615623	-41.654229	89.881755	8.437917	22.584057	-12.430799	2.666589	-27.782604	-70.896321	46.593850	338.106508
1	2.527485	8.348368	-73.278866	47.229129	-14.542264	23.232895	-49.789490	12.271735	64.541538	-19.270709	183.887908
2	-1.088339	92.755476	6.277252	-9.228916	17.676234	10.261368	38.221663	17.556649	36.182406	-32.980740	320.176503
3	-10.926922	-42.551129	-59.293412	-29.879427	-30.658896	-66.262547	-13.541856	80.286776	-51.549295	-70.479207	785.314371
4	-35.949245	-21.577069	54.932978	-33.822514	-18.906987	-4.455433	54.883694	-39.148948	59.607280	-57.340047	3.815126

Principal components

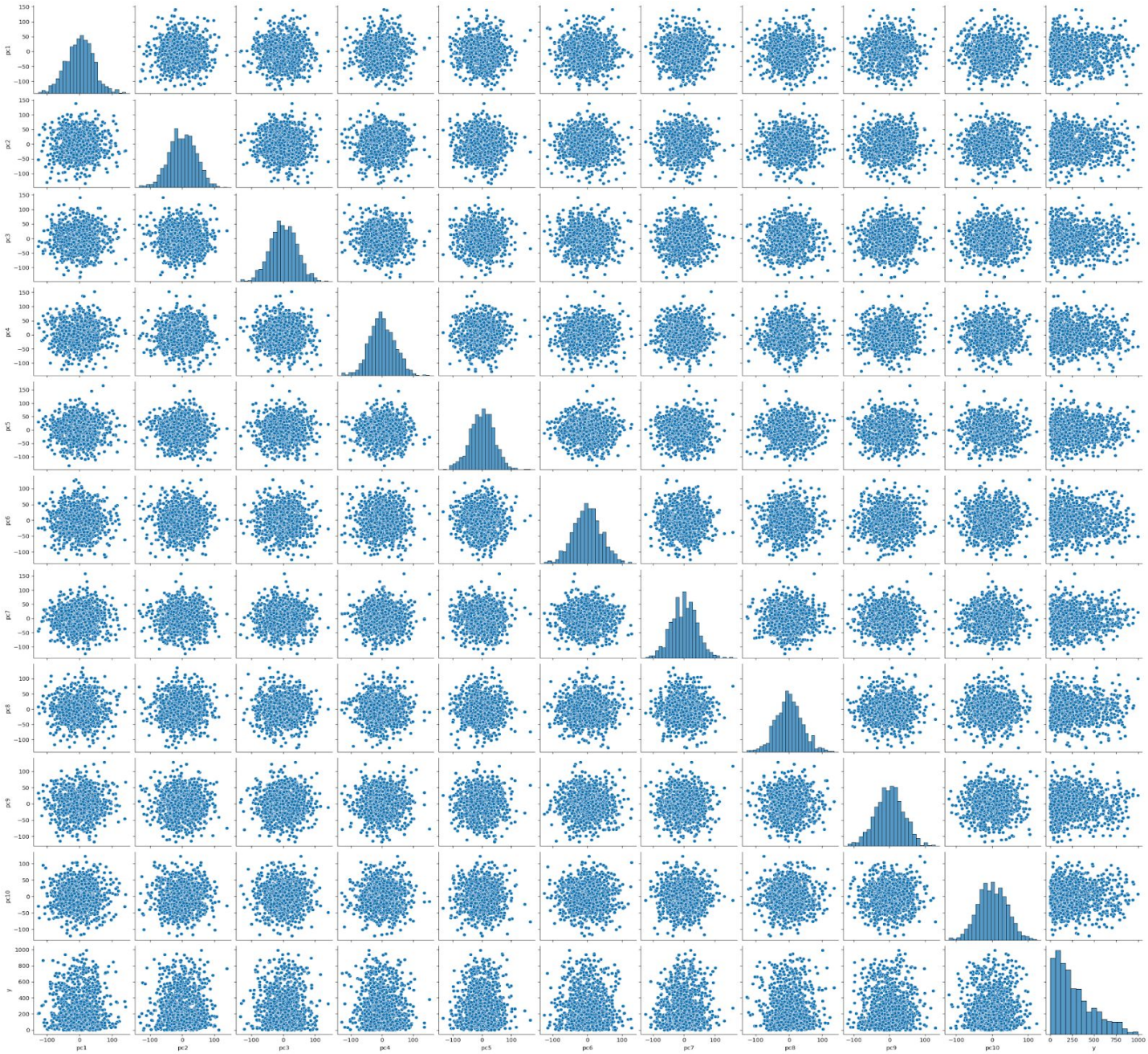
```
plt.figure(figsize=(20, 8))
sns.histplot(df_comp)
```

5.2s Python

```
<Axes: ylabel='Count'>
```



Histogram plot



- On applying EDA on the data we get to know that the data has very poor correlation among the columns. This is leading to larger errors in the regression of the data. The data does not contain any Null values or duplicates in it. X components are nearly of normal distributions on comparing with each Y component in the sns plot.

- **Results for lasso regression**

```
Lasso_MAE,Lasso_MSE,Lasso_RMSE
```
[52] ✓ 0.0s

```
(163.282697205665, 40105.092548836314, 200.48730521895084)
```

- **Results for ridge regression**

```
Ridge_MAE,Ridge_MSE,Ridge_RMSE
```
] ✓ 0.0s

```
(163.40117692095646, 40194.304428783864, 200.4851725908524)
```

- **Results for Linear Regression**

```
MAE,MSE,RMSE
```
40] ✓ 0.0s

```
(163.4026126939647, 40195.15955395675, 200.48730521895084)
```

- **From the results, we get that results for all the data are nearly same but we get better results in the order:**

  lasso>ridge> linear regression.

SVD on the data:

- SVD Algorithm:

1. Singular Value Decomposition (SVD) is a mathematical technique used in linear algebra and numerical analysis. It decomposes a matrix into three simpler matrices, providing insights into the structure and properties of the original matrix.

2. On applying SVD to the given data, the data is decomposed to 50%. We get the data of the same size but decomposed values

```
svd_data
```

Python

	0	1	2	3	4	5	6	7	8	9	...	140	141	142	143	144	145	146	14
0	67.554038	42.593724	72.092361	38.459392	83.982719	38.627386	32.485220	63.651570	76.963950	70.139019	...	48.403576	29.620574	0.249341	49.579308	74.005562	90.794784	7.240388	66.6623
1	31.185371	6.818760	34.987283	49.443128	12.439858	29.462960	35.621607	66.165724	79.255714	9.407669	...	28.328561	62.683684	34.614913	5.724642	72.257690	18.204574	15.875479	40.9070
2	44.028902	94.241313	22.581781	25.322222	66.772974	-2.272033	66.288217	7.709610	64.077417	-7.752297	...	37.415750	43.578757	19.878046	19.782447	58.252627	46.771514	95.797697	31.2374
3	6.664450	12.627522	1.264191	65.911095	51.645416	74.772398	41.554923	6.827582	53.029721	26.905196	...	24.092397	33.214020	14.575806	19.280654	31.876334	21.969677	5.119633	55.3511
4	39.247925	33.569605	69.097506	47.860868	0.262132	43.861449	60.506648	45.261011	105.013635	76.162654	...	7.979324	81.159610	30.721842	76.387630	72.498613	45.317834	66.876848	0.4873
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	24.855140	8.048224	57.866468	22.185148	12.385665	61.879589	23.426887	20.324890	57.008607	10.152460	...	29.854411	-14.797294	43.431164	56.403242	58.661965	28.566920	47.471584	37.1411
996	61.135504	46.909801	50.335444	18.502231	65.770317	37.424814	155.067743	15.125825	18.418905	59.782172	...	56.211298	13.716219	19.969195	15.430680	18.806868	106.739991	66.836322	17.2001
997	80.878420	51.390056	95.768241	13.291327	46.152211	30.358624	93.553591	91.352342	66.911378	59.571336	...	82.538633	11.414285	25.972261	34.434782	94.057109	81.477584	4.382085	-8.6755
998	9.115543	25.238497	27.542861	78.692905	69.297535	63.550282	21.151816	32.672484	14.615721	15.655579	...	37.403446	120.633399	20.503239	57.393188	29.731734	63.709462	80.194922	84.2067
999	23.279549	94.978400	36.908323	69.507544	36.933903	18.849537	39.075495	24.451158	41.408399	72.493382	...	46.953957	20.793559	50.496687	102.006212	101.127886	51.794605	33.839626	20.7499

1000 rows × 150 columns

+ Code + Markdown