Task-D: Collinear features and their effect on linear models

In [79]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
```

In [138]:
```
data = pd.read_csv('task_d.csv')
```

In [139]:
```
data.head()
```

Out[139]:

	x	y	z	x*x	2*y	2*z+3*x*x	w	target
0	-0.581066	0.841837	-1.012978	-0.604025	0.841837	-0.665927	-0.536277	0
1	-0.894309	-0.207835	-1.012978	-0.883052	-0.207835	-0.917054	-0.522364	0
2	-1.207552	0.212034	-1.082312	-1.150918	0.212034	-1.166507	0.205738	0
3	-1.364174	0.002099	-0.943643	-1.280666	0.002099	-1.266540	-0.665720	0
4	-0.737687	1.051772	-1.012978	-0.744934	1.051772	-0.792746	-0.735054	0

In [140]:
```
X = data.drop(['target'], axis=1).values
Y = data['target'].values
```

Doing perturbation test to check the presence of collinearity

Task: 1 Logistic Regression

1. **Finding the Correlation between the features**
    a. check the correlation between the features
    b. plot heat map of correlation matrix using seaborn heatmap
2. **Finding the best model for the given data**
    a. Train Logistic regression on data(X,Y) that we have created in the above cell
    b. Find the best hyper prameter alpha with hyper parameter tuning using k-fold cross validation (grid search CV or
    random search CV make sure you choose the alpha in log space)
    c. Creat a new Logistic regression with the best alpha
    (search for how to get the best hyper parameter value), name the best model as 'best_model'

3. **Getting the weights with the original data**
    a. train the 'best_model' with X, Y
    b. Check the accuracy of the model 'best_model_accuracy'
    c. Get the weights W using best_model.coef_

4. **Modifying original data**
    a. Add a noise(order of 10^-2) to each element of X
    and get the new data set X' (X' = X + e)
    b. Train the same 'best_model' with data (X', Y)
    c. Check the accuracy of the model 'best_model_accuracy_edited'
    d. Get the weights W' using best_model.coef_

5. **Checking deviations in metric and weights**
    a. find the difference between 'best_model_accuracy_edited' and 'best_model_accuracy'
    b. find the absolute change between each value of W and W' ==> |(W-W')|
    c. print the top 4 features which have higher % change in weights
    compare to the other feature

Task: 2 Linear SVM

1. Do the same steps (2, 3, 4, 5) we have done in the above task 1.

Do write the observations based on the results you get from the deviations of weights in both Logistic Regression and linear SVM

Task 1:

Logistic Regression:

Correlation between features:

```
In [141]: data.corr()
```

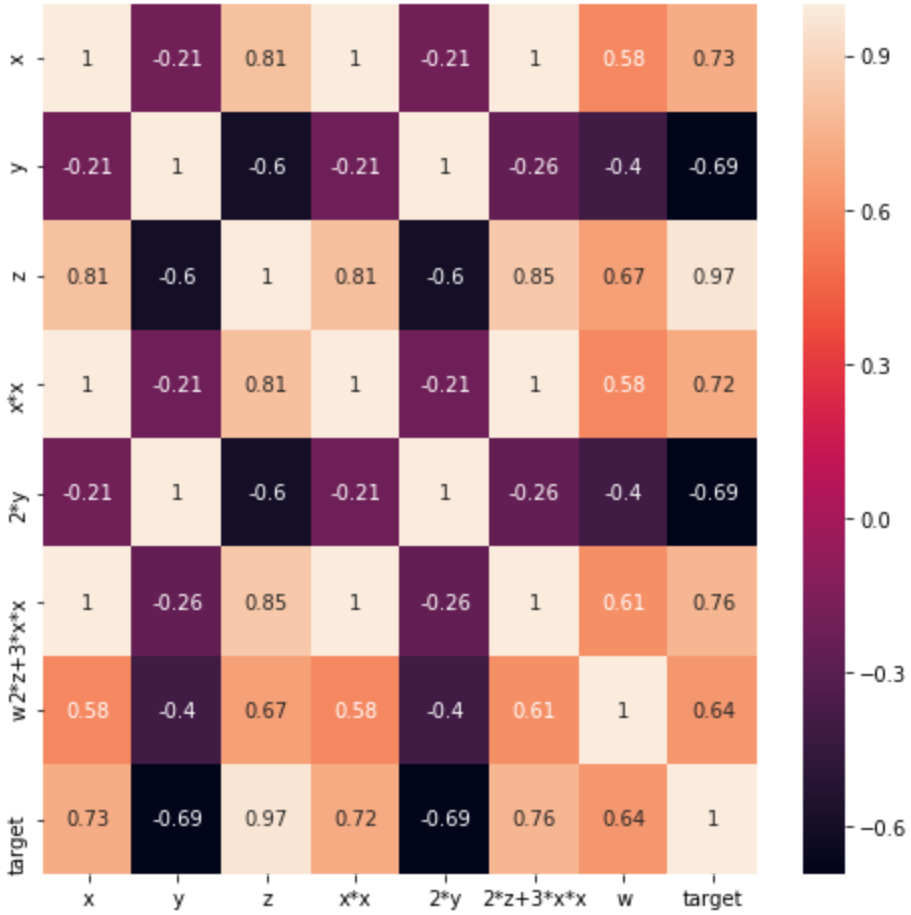
Out[141]:

	x	y	z	x*x	2*y	2*z+3*x*x	w	target
x	1.000000	-0.205926	0.812458	0.997947	-0.205926	0.996252	0.583277	0.728290
y	-0.205926	1.000000	-0.602663	-0.209289	1.000000	-0.261123	-0.401790	-0.690684
z	0.812458	-0.602663	1.000000	0.807137	-0.602663	0.847163	0.674486	0.969990
x*x	0.997947	-0.209289	0.807137	1.000000	-0.209289	0.997457	0.583803	0.719570
2*y	-0.205926	1.000000	-0.602663	-0.209289	1.000000	-0.261123	-0.401790	-0.690684
2*z+3*x*x	0.996252	-0.261123	0.847163	0.997457	-0.261123	1.000000	0.606860	0.764729
w	0.583277	-0.401790	0.674486	0.583803	-0.401790	0.606860	1.000000	0.641750
target	0.728290	-0.690684	0.969990	0.719570	-0.690684	0.764729	0.641750	1.000000

Heat map of correlation matrix:

In [142]:
```
plt.figure(figsize=(8,8))
sns.heatmap(data.corr(),annot=True)
```

Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x49528320>



Finding the best model for the data :

In [143]:
```
lr=SGDClassifier(loss='log',random_state=15)
params={'alpha':[0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]}
clf=GridSearchCV(lr,params,cv=5)
clf.fit(X,Y)
clf.best_params_
```

Out[143]: {'alpha': 0.001}

In [144]:
```
best_model=SGDClassifier(loss='log',alpha=0.001,random_state=15)
best_model.fit(X,Y)
best_model
```

Out[144]: SGDClassifier(alpha=0.001, loss='log', random_state=15)

In [145]:
```
from sklearn.metrics import accuracy_score
y_pred=best_model.predict(X)
best_model_accu=accuracy_score(Y,y_pred)
best_model_accu
```

Out[145]: 1.0

In [146]:
```
w=best_model.coef_
print('weights:',w)
```

```
weights: [[ 1.4731831  -1.36089554  3.35194144  1.27509445 -1.36089554  1.552283
97
   1.35689461]]
```

Adding noise to the dataset and applying Logistic regression:

In [147]:
```
X_new = [i + 0.01 for i in X]
```

In [148]:
```
best_model_edited=SGDClassifier(loss='log',alpha=0.001,random_state=15)
best_model_edited.fit(X_new,Y)
best_model_edited
```

Out[148]:
```
SGDClassifier(alpha=0.001, loss='log', random_state=15)
```

In [149]:
```
y_pred=best_model_edited.predict(X_new)
best_model_accuracy_edited=accuracy_score(Y,y_pred)
best_model_accuracy_edited
```

Out[149]:
```
1.0
```

In [150]:
```
w_edit=best_model_edited.coef_
print('weights for the edited dataset:\n',w_edit)
```

```
weights for the edited dataset:
 [[ 1.45499505 -1.35274634  3.34051158  1.25874152 -1.35274634  1.53613683
   1.33612048]]
```

In [151]:
```
accuracy_diff=best_model_accuracy_edited-best_model_accu
print('Difference between best_model_accuracy_edited and best_model_accuracy:\n
',accuracy_diff)
```

```
Difference between best_model_accuracy_edited and best_model_accuracy:
 0.0
```

In [152]:
```
weight_deviation=np.abs(w-w_edit)
print('absolute change between each value of W and W:\n',weight_deviation)
```

```
absolute change between each value of W and W:
 [[0.01818805 0.0081492  0.01142986 0.01635294 0.0081492  0.01614714
  0.02077412]]
```

In [153]:
```
print("Top 4 features which have higher % change in weights compare to the othe
r features:")
columns = list(data.columns)
for index in np.argsort(-weight_deviation).reshape(-1)[:4]:
    print(columns[index])
```

```
Top 4 features which have higher % change in weights compare to the other featur
es:
w
x
x*x
2*z+3*x*x
```

Task-2

Linear SVM

Finding the best model for the data:

In [154]:
```
svm=SGDClassifier(loss='hinge',random_state=15)
params={'alpha':[0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]}
clf=GridSearchCV(svm,params,cv=5)
clf.fit(X,Y)
clf.best_params_
```

Out[154]:
```
{'alpha': 1e-05}
```

In [155]:
```
best_model_svm=SGDClassifier(loss='hinge',alpha=0.00001,random_state=15)
best_model_svm.fit(X,Y)
best_model_svm
```

Out[155]:
```
SGDClassifier(alpha=1e-05, random_state=15)
```

In [156]:
```
from sklearn.metrics import accuracy_score
y_pred1=best_model_svm.predict(X)
best_model_accu_svm=accuracy_score(Y,y_pred1)
best_model_accu_svm
```

Out[156]:
```
1.0
```

In [157]:
```
w_svm=best_model_svm.coef_
print('weights:',w_svm)
```

```
weights: [[ 24.77242963 -23.24335792  49.349105    22.06653062 -23.24335792
   25.81810632  22.0688594 ]]
```

Adding noise to the dataset and applying SVM:

In [158]:
```
X_ne = [i + 0.01 for i in X]
```

In [159]:
```
best_model_edited_svm=SGDClassifier(loss='hinge',alpha=0.00001,random_state=15)
best_model_edited_svm.fit(X_ne,Y)
best_model_edited_svm
```

Out[159]:
```
SGDClassifier(alpha=1e-05, random_state=15)
```

In [160]:
```
y_pred=best_model_edited_svm.predict(X_new)
best_model_accuracy_edited_svm=accuracy_score(Y,y_pred)
best_model_accuracy_edited_svm
```

Out[160]:
```
1.0
```

In [161]:
```
w_edit_svm=best_model_edited_svm.coef_
print('weights:',w_edit_svm)
```

```
weights: [[ 24.77242963 -23.24335792  49.349105    22.06653062 -23.24335792
   25.81810632  22.0688594 ]]
```

In [162]:
```
accuracy_diff_svm=best_model_accuracy_edited_svm-best_model_accu_svm
print('Difference between best_model_accuracy_edited and best_model_accuracy:\n
',accuracy_diff_svm)
```

```
Difference between best_model_accuracy_edited and best_model_accuracy:
 0.0
```

In [163]:
```
weight_deviation_svm=np.abs(w_svm-w_edit_svm)
print('absolute change between each value of W and W:\n',weight_deviation_svm)
```

```
absolute change between each value of W and W:
 [[0.00000000e+00 3.55271368e-15 0.00000000e+00 0.00000000e+00
  3.55271368e-15 0.00000000e+00 0.00000000e+00]]
```

In [164]:
```
print("Top 4 features which have higher % change in weights compare to the othe
r features:")
columns = list(data.columns)
for index in np.argsort(-weight_deviation_svm).reshape(-1)[:4]:
    print(columns[index])
```

```
Top 4 features which have higher % change in weights compare to the other featur
es:
y
2*y
x
z
```

Observation:

Some of the features are highly corelated as seen from the correlation matrix.

After applying both Logistic Regression and Linear SVM on the dataset and performing perturbation tests to check the presence of collinearity,we can notice the percentage change in the estimates of original data and perturbed data is very small.There is very small change in the weights after applying both the models.

Hence the collinearity is relatively stable or no collinearity can be observed on the dataset.