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

In [228]:

```
from google.colab import drive
drive.mount('/gdrive')
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdrive", forc
e_remount=True).
```

In [229]:

```
path='/gdrive/My Drive/AAIC/8_LinearModels-20200808T170255Z-001/8_LinearModels/'
```

In [230]:

```
%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 [231]:

```
data = pd.read_csv(path+'task_d.csv')
data.shape
```

Out[231]:

```
(100, 8)
```

In [232]:

```
data.head()
```

Out[232]:

	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 [233]:

```
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

In [234]:

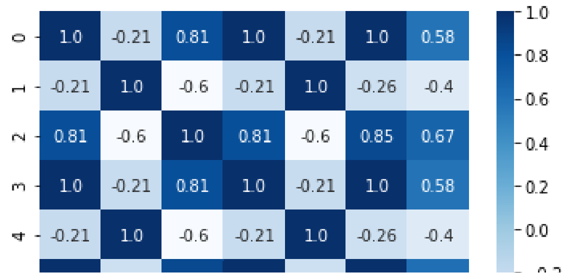
```
corr_mat = pd.DataFrame(np.corrcoef(X,rowvar=False))
#print(corr_mat)
```

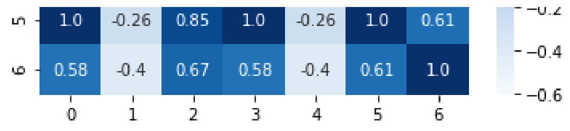
In [235]:

```
#from sklearn.preprocessing import StandardScaler
#X = StandardScaler().fit_transform(X)
```

In [236]:

```
sns.heatmap(corr_mat,annot=True,fmt='.2',cmap='Blues')
plt.show()
```





In [237]:

```
alpha = np.logspace(-4,4)
#print(alpha)
hyper_param = {'alpha':alpha }

clf = SGDClassifier(loss='log')
model = GridSearchCV(clf,hyper_param,scoring='accuracy',return_train_score=True)
model.fit(X,Y)

results = pd.DataFrame.from_dict(model.cv_results_)
results = results.sort_values(['param_alpha'])

results.head()
```

Out[237]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_score	split1_test_s
0	0.000694	0.000129	0.000292	0.000029	0.0001	{'alpha': 0.0001}	1.0	
1	0.000646	0.000046	0.000316	0.000061	0.000145635	{'alpha': 0.00014563484775012445}	1.0	
2	0.000603	0.000020	0.000273	0.000013	0.000212095	{'alpha': 0.00021209508879201905}	1.0	
3	0.000624	0.000019	0.000269	0.000011	0.000308884	{'alpha': 0.00030888435964774815}	1.0	
4	0.000582	0.000009	0.000283	0.000038	0.000449843	{'alpha': 0.0004498432668969444}	1.0	

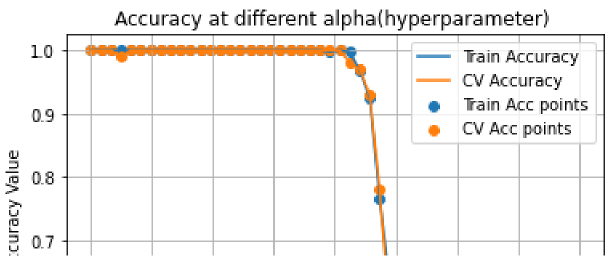
In [238]:

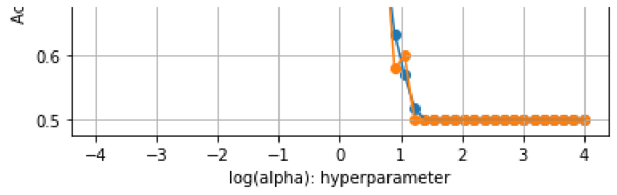
```
train_acc= results['mean_train_score']
train_acc_std= results['std_train_score']
cv_acc = results['mean_test_score']
cv_acc_std= results['std_test_score']
a =  results['param_alpha']
log_a = np.log10(list(a))
#print(a)
plt.plot(log_a, train_acc, label='Train Accuracy')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(log_a, cv_acc, label='CV Accuracy')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(log_a, train_acc, label='Train Acc points')
plt.scatter(log_a, cv_acc, label='CV Acc points')

plt.legend()
plt.grid(b=True)
plt.title("Accuracy at different alpha(hyperparameter) ")
plt.xlabel("log(alpha): hyperparameter")
plt.ylabel("Accuracy Value")
plt.show()
```





In [239]:

```
print(model.best_score_,model.best_params_)
best_alpha = (model.best_params_['alpha'])
print(best_alpha)
```

```
1.0 {'alpha': 0.0001}
0.0001
```

In [240]:

```
from sklearn.metrics import accuracy_score

best_model = SGDClassifier(loss='log',alpha=best_alpha)
best_model.fit(X,Y)
w_lr = best_model.coef_
b_lr = best_model.intercept_
print("Best model coefficients and intercept are",w_lr,b_lr)
Y_predict = best_model.predict(X)
best_model_accuracy = accuracy_score(Y,Y_predict)
print("Best model Accuracy is",best_model_accuracy)
```

```
Best model coefficients and intercept are [[  7.93607591 -10.41754733  17.69006633   7.13278597 -1
0.41754733
    8.55533348   9.09743635]] [-4.82060045]
Best model Accuracy is 1.0
```

Pertubation Test

In [241]:

```
#print(X[:2])
noise = (np.random.randn(X.shape[0],X.shape[1]))*(10**-2)
#print(noise[:2])
X_mod = X+noise
#print(X_mod[:2])
```

In [242]:

```
from sklearn.metrics import accuracy_score

best_model.fit(X_mod,Y)
w_mod=best_model.coef_
b_mod=best_model.intercept_
print("Best model coefficients and intercept after adding noise are",w_mod,b_mod)
Y_predict_mod = best_model.predict(X_mod)
best_model_accuracy_edited = accuracy_score(Y,Y_predict_mod)
print("Best model accuracy after pertubation is",best_model_accuracy_edited)
```

```
Best model coefficients and intercept after adding noise are [[ 5.82770429 -6.93848381  8.98287053
4.88954087 -6.83900899  5.63204005
   0.44640223]] [1.67545911]
Best model accuracy after pertubation is 1.0
```

In [243]:

```
print("Difference in accuracy before and after pertubation test is {}%".format(best_model_accuracy
- best_model_accuracy_edited))
#print(w)
#print(w_mod)
print(abs(w-w_mod))
```

```
#print(abs((w-w_mod)/(w))*100)
features=['x','y','z','x*x','2*y','2*z+3*x*x','w']
pcg_error = [i for i in (abs((w-w_mod)/(w))*100)[0]]
print("% changes of coefficients are",pcg_error)
ftr_dict = dict((i,j) for i,j in zip(features, pcg_error))
top_features = sorted(ftr_dict.items(), key = lambda x:(x[1],x[0]),reverse = True)
no_top_ftr = 4
top4_ftr = [top_features[i][0] for i in range(no_top_ftr)]
print("Top 4 features that are more impacted by pertubation test are",top4_ftr)
```

```
Difference in accuracy before and after pertubation test is 0.0%
[[ 4.11955727  4.28240808 10.33158322  4.83516429  4.3818829   5.45216052
   2.89999945]]
% changes of coefficients are [41.4139835765838, 38.16459619899209, 53.491459567339916,
49.72042043748889, 39.051110607721036, 49.18857689318631, 118.19378606186572]
Top 4 features that are more impacted by pertubation test are ['w', 'z', 'x*x', '2*z+3*x*x']
```

Task 2

In [244]:

```
alpha = np.logspace(-4,4)
#print(alpha)
hyper_param = {'alpha':alpha }

clf = SGDClassifier(loss='hinge')
model = GridSearchCV(clf,hyper_param,scoring='accuracy',return_train_score=True)
model.fit(X,Y)

results = pd.DataFrame.from_dict(model.cv_results_)
results = results.sort_values(['param_alpha'])

results.head()
```

Out[244]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_score	split1_test_s
0	0.000920	0.000481	0.000364	0.000154	0.0001	{'alpha': 0.0001}	1.00	
1	0.000538	0.000022	0.000274	0.000018	0.000145635	{'alpha': 0.00014563484775012445}	1.00	
2	0.000524	0.000026	0.000252	0.000010	0.000212095	{'alpha': 0.00021209508879201905}	0.95	
3	0.000512	0.000012	0.000253	0.000011	0.000308884	{'alpha': 0.00030888435964774815}	1.00	
4	0.000515	0.000011	0.000260	0.000017	0.000449843	{'alpha': 0.0004498432668969444}	1.00	

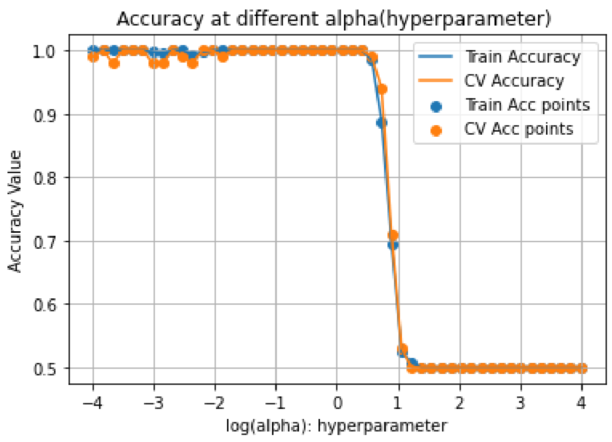
In [245]:

```
train_acc= results['mean_train_score']
train_acc_std= results['std_train_score']
cv_acc = results['mean_test_score']
cv_acc_std= results['std_test_score']
a =  results['param_alpha']
log_a = np.log10(list(a))
#print(a)
plt.plot(log_a, train_acc, label='Train Accuracy')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(log_a, cv_acc, label='CV Accuracy')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(log_a, train_acc, label='Train Acc points')
plt.scatter(log_a, cv_acc, label='CV Acc points')

plt.legend()
plt.grid(b=True)
plt.title("Accuracy at different alpha(hyperparameter) ")
plt.xlabel("log(alpha): hyperparameter")
plt.ylabel("Accuracy Value")
plt.show()
```



In [246]:

```
print(model.best_score_,model.best_params_)
best_alpha = (model.best_params_['alpha'])
print(best_alpha)
```

```
1.0 {'alpha': 0.00014563484775012445}
0.00014563484775012445
```

In [247]:

```
from sklearn.metrics import accuracy_score

best_model = SGDClassifier(loss='log',alpha=best_alpha)
best_model.fit(X,Y)
w_svm = best_model.coef_
b_svm = best_model.intercept_

print("Best model coefficients and intercept are",w_svm,b_svm)
Y_predict = best_model.predict(X)
best_model_accuracy = accuracy_score(Y,Y_predict)
print("Best model Accuracy is",best_model_accuracy)
```

```
Best model coefficients and intercept are [[ 5.45830585 -5.48842096 14.01207218  4.50567903 -5.488
42096  5.74686071
   5.50559227]] [6.12407267]
Best model Accuracy is 1.0
```

Pertubation Test

In [248]:

```
#print(X[:2])
noise = (np.random.randn(X.shape[0],X.shape[1]))*(10**-2)
#print(noise[:2])
X_mod = X+noise
#print(X_mod[:2])
```

In [249]:

```
from sklearn.metrics import accuracy_score

best_model.fit(X_mod,Y)
w_mod = best_model.coef_
b_mod = best_model.intercept_
print("Best model coefficients and intercept after pertubation are",w_mod,b_mod)
```

```
Y_predict_mod = best_model.predict(X_mod)
best_model_accuracy_edited = accuracy_score(Y,Y_predict_mod)
print("Best model accuracy after pertubation is",best_model_accuracy_edited)
```

```
Best model coefficients and intercept after pertubation are [[ 3.27282183 -8.52740834 15.25662712
2.69024397 -8.45696979  4.35628708
   1.59529907]] [-3.92829642]
Best model accuracy after pertubation is 1.0
```

In [250]:

```
print("Difference in accuracy before and after pertubation test is {}%".format(best_model_accuracy
- best_model_accuracy_edited))
#print(w)
#print(w_mod)
print("Absolute difference in coefficients before and after pertubation is",abs(w-w_mod))
#print(abs((w-w_mod)/(w))*100)
features=['x','y','z','x*x','2*y','2*z+3*x*x','w']
pcg_error = [i for i in (abs((w-w_mod)/(w))*100)[0]]
print("% changes in error of coefficients are",pcg_error)
ftr_dict = dict((i,j) for i,j in zip(features, pcg_error))
top_features = sorted(ftr_dict.items(), key = lambda x:(x[1],x[0]),reverse = True)
no_top_ftr = 4
top4_ftr = [top_features[i][0] for i in range(no_top_ftr)]
print("Top 4 features that are more impacted by pertubation test are",top4_ftr)
```

```
Difference in accuracy before and after pertubation test is 0.0%
Absolute difference in coefficients before and after pertubation is [[6.67443973 2.69348355
4.05782662 7.03446119 2.7639221  6.72791349
  4.04889629]]
% changes in error of coefficients are [67.09826303634637, 24.004184142306848, 21.009274595109158,
72.33598418056393, 24.631928791796383, 60.6982294248896, 165.0187838031198]
Top 4 features that are more impacted by pertubation test are ['w', 'x*x', 'x', '2*z+3*x*x']
```

Observations from the above two Tasks.

1. Percentage changes in the coefficients of the model after pertubation test are more in LR than svm. This implies that svm tries to fit less to the noise data than LR. So the model does not change by large extent even though we used same regularization parameter(α).
2. But in both LR and svm 'w' is the feature which has high % change in coefficients. **Conclusion:** From the above assignment, we can conclude that for collinear features svm is better model than LR as it has less % change in coefficients of the model.