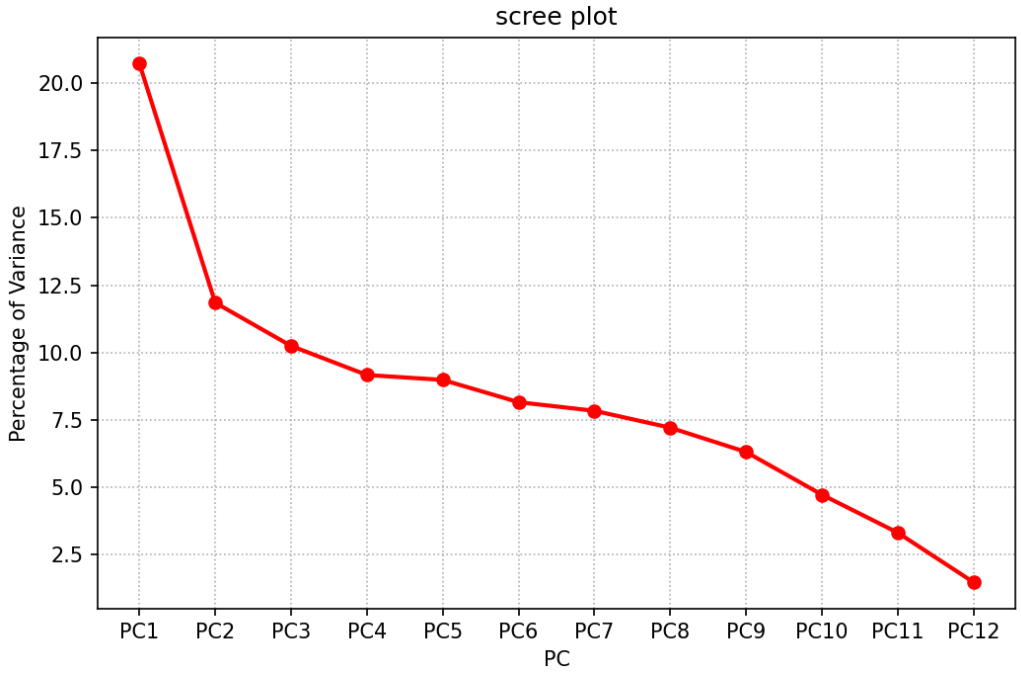
Data preprocessing:

```
file_path = 'H:\\tamu\\courses\\MSEN660\\Homeworks\\computer_project3\\'
SMA_org = pd.read_csv(file_path + 'Soft_Magnetic_Alloy_Dataset.csv')

# data pre processing
features_org = SMA_org.columns[0:26]
feature_val_org = SMA_org.values[:, 0:26]
response_org = SMA_org['Coercivity (A/m)']   # select response

num_samples = feature_val_org.shape[0]
feature_value_sum = np.sum(feature_val_org > 0, axis=0) / num_samples
is_selected = feature_value_sum > 0.05      # true/false  --> condition on column

# filtering data
feature_val_filt = feature_val_org[:, is_selected]      # get features with more than 5% feature val sum
no_NAN = np.invert(np.isnan(response_org))  # true/false --> all rows with real value of response --> condition on row
feature_val = feature_val_filt[no_NAN, :]
response = response_org[no_NAN]
features_name = features_org[is_selected]

# add random perturbation to the features: adding zero mean Gaussian noise to feature value
np.random.seed(0)
std = 2
mean = 0
n, d = feature_val.shape
noise = np.random.normal(mean, std, [n, d])
feature_val_noisy = feature_val + noise
feature_val_noisy = (feature_val_noisy + abs(feature_val_noisy)) / 2    # clamp values at zero (?????)

# normalize data
feature_val_normalized = ssc().fit_transform(feature_val_noisy)
```

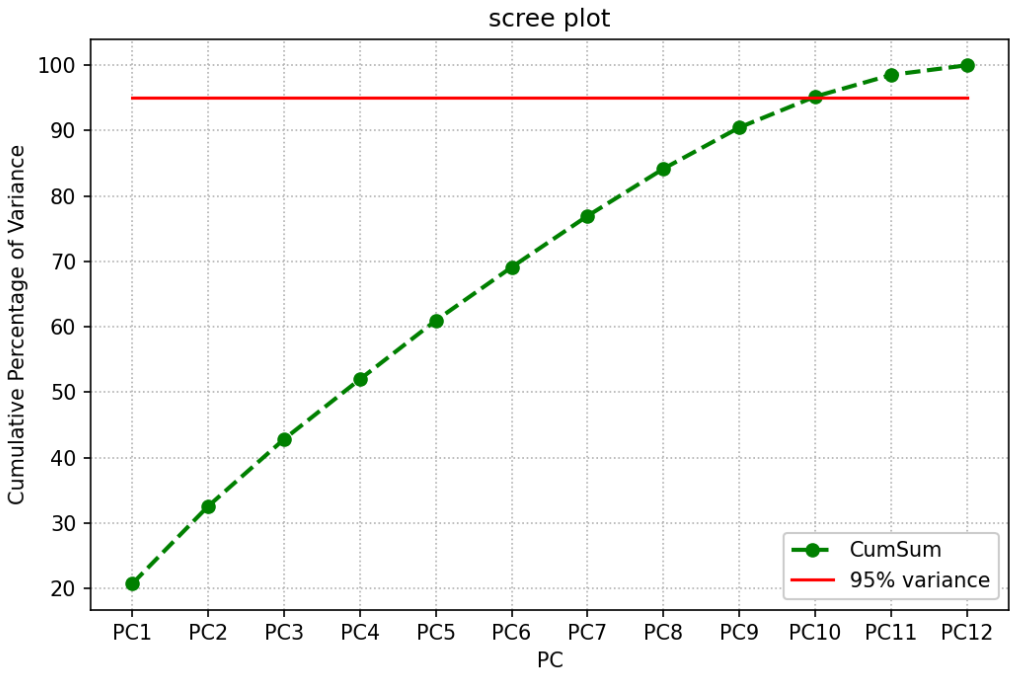
Running PCA and plot “scree plot”:

```
# compute PCA
pca = PCA().fit(feature_val_normalized)
variance = pca.explained_variance_ratio_
cumsum = np.cumsum(variance)

components = ['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12']

# scree plot
def get_scree_plot(data, componnets ,ltype, xlabel, ylabel, figname):
    plt.figure(figsize=(8, 5))
    x = np.arange(start=0, stop=len(componnets), step=1)
    plt.xticks(x, componnets)
    plt.plot(100 * data, ltype, linewidth=2)  # 'ro-'
    plt.xlabel(xlabel) #'PC'
    plt.ylabel(ylabel)    #'Percentage of Variance'
    plt.title('scree plot')
    plt.grid(linestyle='dotted')
    figpath = file_path + figname + '.png'    # 'Variance'
    plt.savefig(figpath, dpi=150)


get_scree_plot(variance, components, 'ro-', 'PC', 'Percentage of Variance', 'Variance')
```

Data varaince explained by PC:



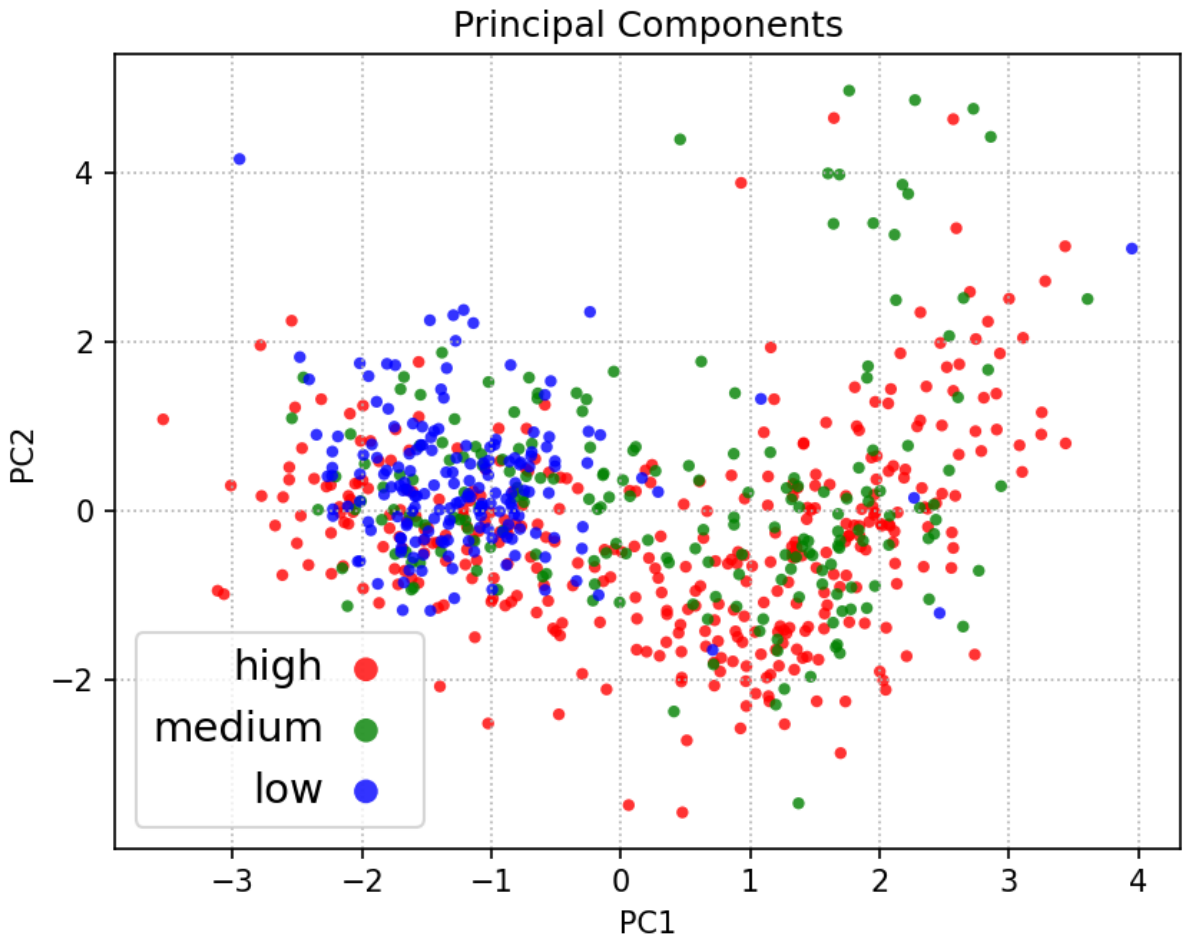
Variance accumulative:

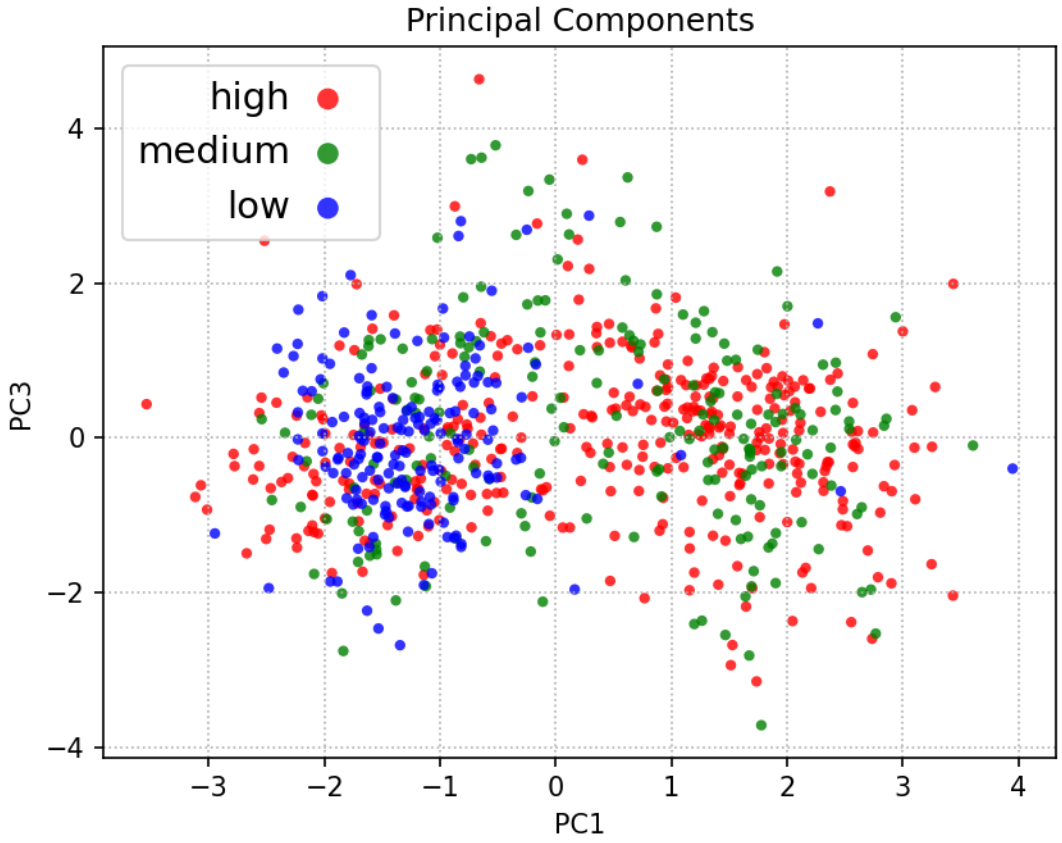


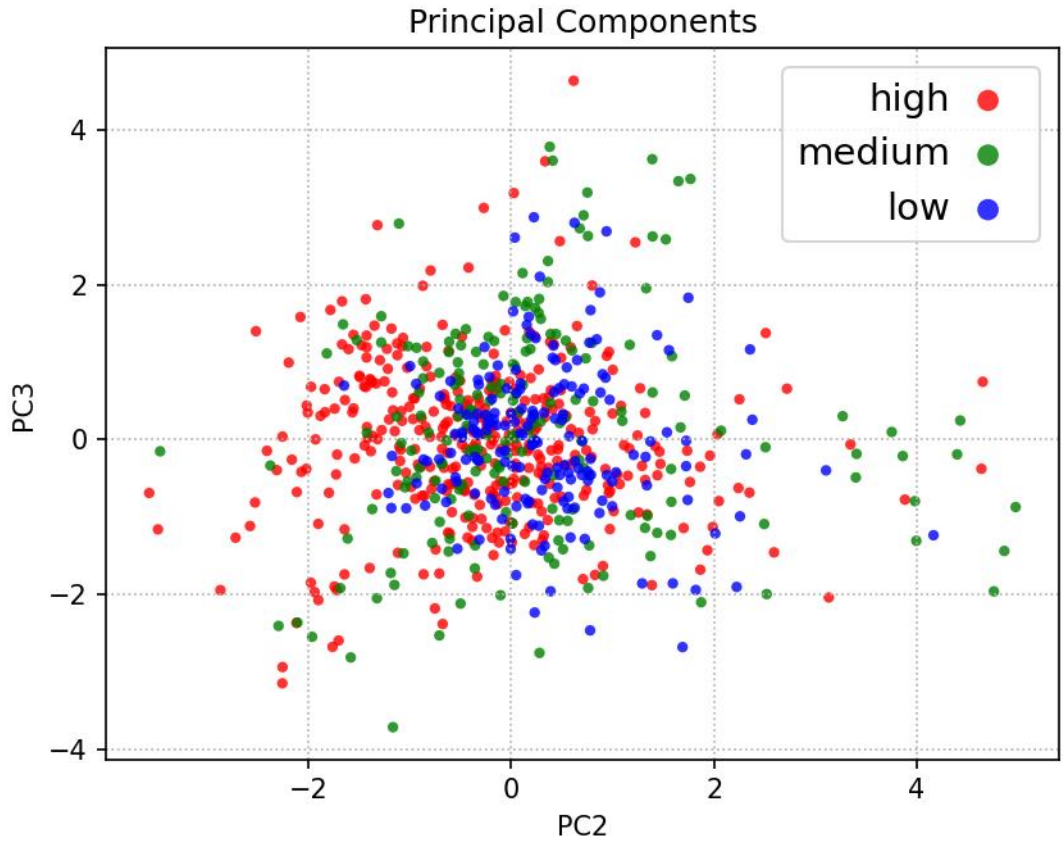
We need 10 PC (PC1 to PC10) to explain 95% of the data variance

Scatterplots:

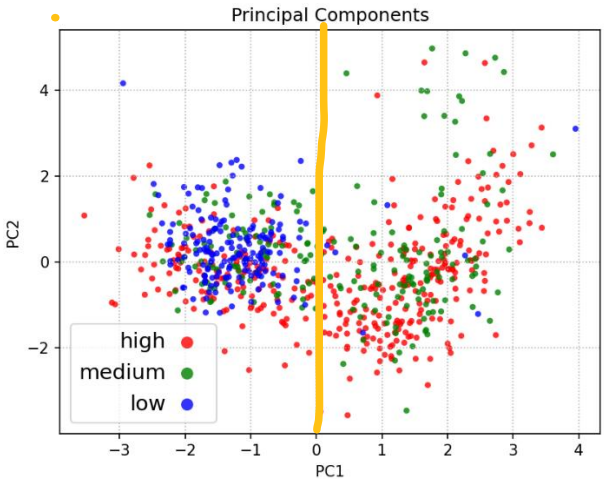
```
def get_scatter_plot(X_pca, components, xlabel, ylabel):
    FirstPC = components.index(xlabel)
    SecondPC = components.index(ylabel)

    plt.figure()
    plt.scatter(X_pca[high_ind, FirstPC], X_pca[high_ind, SecondPC], alpha=0.8, c='red', edgecolors='none', s=16, label='high')
    plt.scatter(X_pca[medium_ind, FirstPC], X_pca[medium_ind, SecondPC], alpha=0.8, c='green', edgecolors='none', s=16,label='medium')
    plt.scatter(X_pca[low_ind, FirstPC], X_pca[low_ind, SecondPC], alpha=0.8, c='blue', edgecolors='none', s=16, label='low')
    plt.legend(fontsize=14, facecolor='white', markerscale=2, markerfirst=False, handletextpad=0)
    plt.grid(linestyle='dotted')
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title('Principal Components')
    plt.xticks(size='medium')
    plt.yticks(size='medium')
    figname = xlabel + '_vs_' + ylabel
    plt.savefig(file_path + figname + '.png',dpi=150)


components = ['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12']
get_scatter_plot(x_pca, components, 'PC1', 'PC2')
get_scatter_plot(x_pca, components, 'PC1', 'PC3')
get_scatter_plot(x_pca, components, 'PC2', 'PC3')
```




Principal Components
high
medium
low
PC3
PC1
4
2
0
−2
−4
−3
−2
−1
0
1
2
3
4


Principal Components
high
medium
low
PC3
PC2
4
2
0
−2
−4
−2
0
2
4

Based on the plots, PC1 (which explain 20% of the variance of the data) shows the maximum discrimination over the data. The first plot (Pc1vs PC2) could have a linear classifier which discriminate between low vs. high/medium. For example:



So if we want to choose only one PC, we would choose PC1.

Loading Matrix:

```
loading = pca.components_.T *np.sqrt(pca.explained_variance_)
loading_matrix = pd.DataFrame(loading, columns=components, index=features_name)
```

Printing the loading matrix:

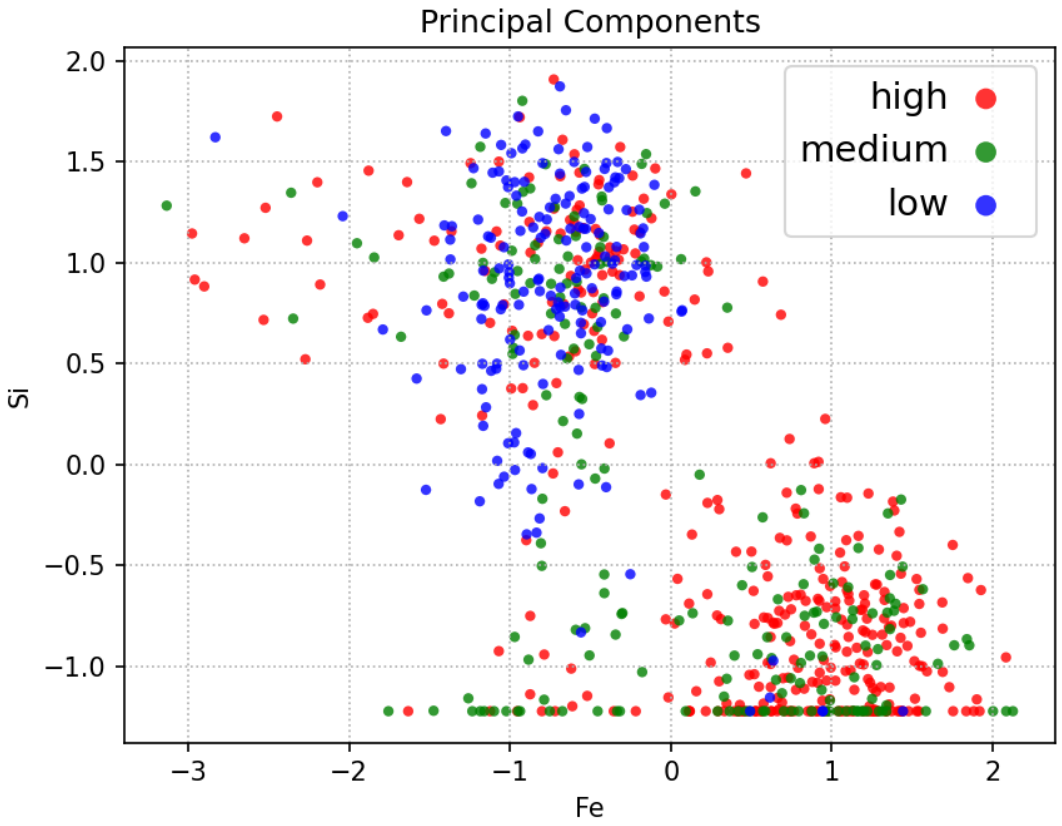
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12
Fe	0.781706	-0.43077	-0.01132	0.020025	-0.02129	-0.01672	-0.01625	-0.0432	0.166697	-0.18797	-0.2872	-0.23607
Si	-0.82663	0.138913	-0.18146	-0.21953	-0.16803	-0.08912	-0.15075	-0.00767	-0.22045	0.161648	-0.01735	-0.28917
Al	-0.01299	0.597578	-0.17058	0.151762	0.326764	-0.07698	-0.09345	0.620622	0.243423	-0.02722	-0.15642	-0.02976
B	-0.3474	-0.51053	0.409058	-0.23955	0.067467	0.016302	0.233115	0.442324	0.234746	-0.11058	0.261754	-0.0682
P	0.662175	0.536243	-0.07768	-0.13799	0.101508	-0.02771	0.031485	-0.18749	0.062049	-0.13812	0.40241	-0.13478
Ge	-0.04717	0.196093	0.54205	0.62503	0.045383	0.185242	0.247893	0.047629	-0.3946	-0.1091	-0.00968	-0.09161
Cu	-0.00695	0.173517	0.385612	-0.02763	-0.34592	0.570821	-0.56543	0.004269	0.237532	0.007049	0.004831	-0.00427
Zr	0.201185	-0.32247	-0.4387	0.622646	-0.33049	-0.01269	-0.09868	0.199679	0.088163	0.24606	0.219651	-0.03444
Nb	-0.48402	0.122921	0.196596	0.319929	0.150556	-0.15281	0.18553	-0.42864	0.569132	0.13947	-0.03207	-0.04892
Mo	-0.08435	0.108114	-0.40582	-0.1231	-0.10399	0.673393	0.570679	0.00261	0.0883	0.0293	-0.06159	-0.01724
W	0.077316	-0.29818	-0.05042	0.016678	0.79986	0.339704	-0.23368	-0.08205	-0.10727	0.265895	0.058344	-0.03891
Anne	-0.58849	-0.15954	-0.39339	0.237558	0.158545	0.111735	-0.22197	-0.12661	0.021061	-0.55749	0.063293	0.015093

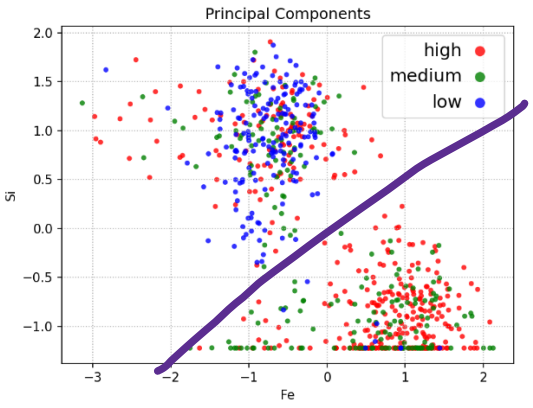
The absolute value of the coefficients indicate the relative importance of each original variable

(row of W) in the corresponding PC (column of W). ➔ in PC1, ‘Fe’ and ‘Si’ has the biggest absolute value of coefficient that means they explain most of the variance of the data and contribute the discriminating PC (PC1).

This method help us in feature selection.

Plotting data based on ‘Fe’ , and ‘Si’





We visually see that a linear classifier can approximately discriminate the data into low versus medium/high coercivity.
Although the medium and high coercivity cannot be classified through only Si and Fe