Dimentionality Reduction

Lighthouse Labs
September 24, 2020

Instructor: Socorro Dominguez

Today's Agenda

1. Why do we need dimensionality reduction? (*5 min*)
2. Different Techniques
    - Variable Selection (*10 min*)
        - Filter Methods
        - Wrapper Methods
    - Dimensionality Reduction (*10 min*)
        - PCA
3. Questions (*5 min*)

What is Dimentionality Reduction?

Dimentionality Reduction is discovering key features in the data or relationships in the data that allow us to reduce the number of variables we are working with.

If we had a 10 features space, but we could reduce it to a just 2 or 3 variables space, we could even visualize it.

Case Study

Suppose we all measured all of Lighthouse Labs students' heights and we got the next set:

```
In [8]: # Generating a set of heights from a Normal distribution with mean 1.7 and std = .
        1
        heights = np.random.normal(1.7, scale=.05, size=15)
        heights
```

```
Out[8]: array([1.66378233, 1.65543962, 1.81282448, 1.70099206, 1.75762911,
               1.65419298, 1.61485147, 1.6764591 , 1.68713196, 1.7160945 ,
               1.72204351, 1.69181278, 1.71523349, 1.65190621, 1.81540095])
```

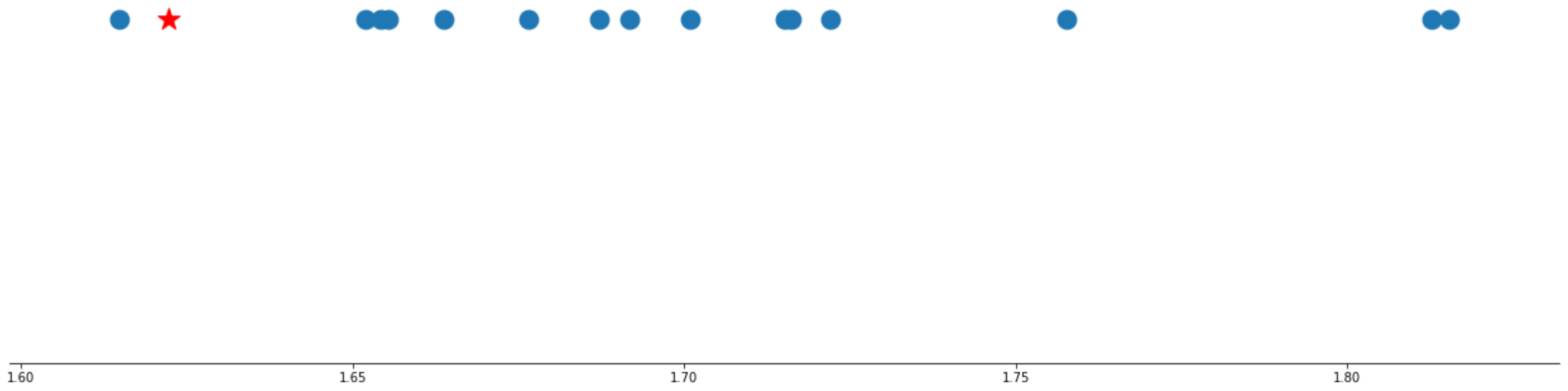
- Then, I ask you: "How tall are Lighthouse Labs students?"

Well, maybe you know that if you give me all these numbers, I will forget them.

So you think: "What single number best represents all these heights that I have?" We could use any, right? But maybe we would want one that represents all numbers best.

What makes you pick one point over any other to be the red star?

In [11]:
```
plot_random_summary(22,10)
```

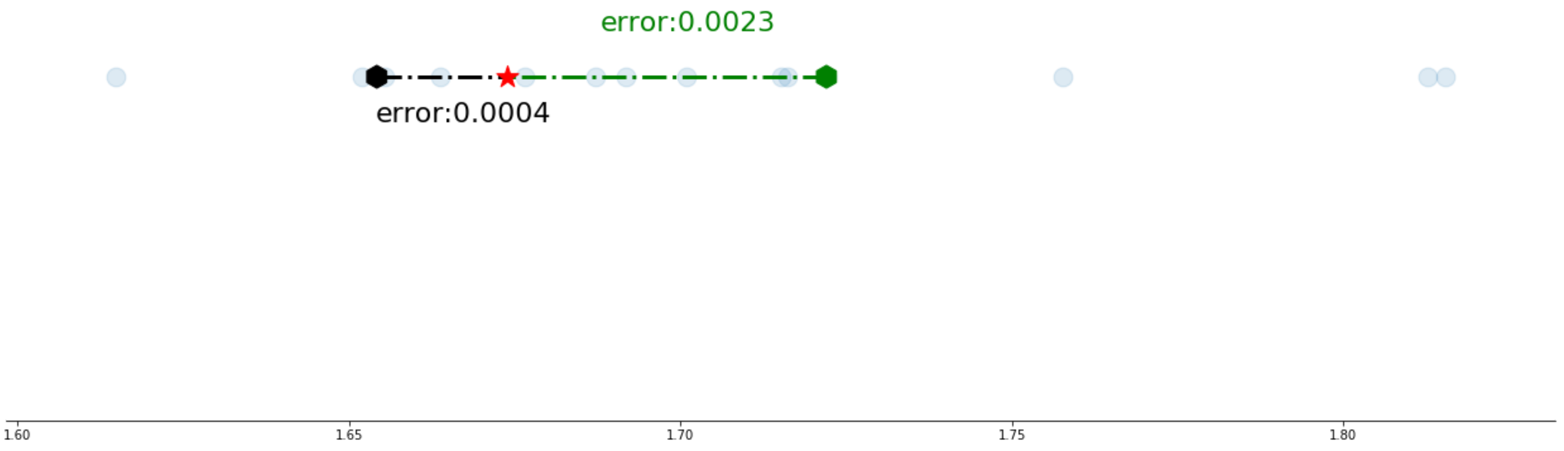


But... Wouldn't we lose information if we reduced data?

By using only one number to represent all the heights we have in the dataset, we will be losing information. We can measure how wrong we are by defining an error measure.

For example, we could use $(x_i - p)^2$ as the error of using p to represent x_i

In [13]:
```
plot_summary_error(22,10)
```



So you tell me: "The average height of the Lighthouse Labs students is 1.7m"

But what just happened? You reduced 15 numbers down to only one.

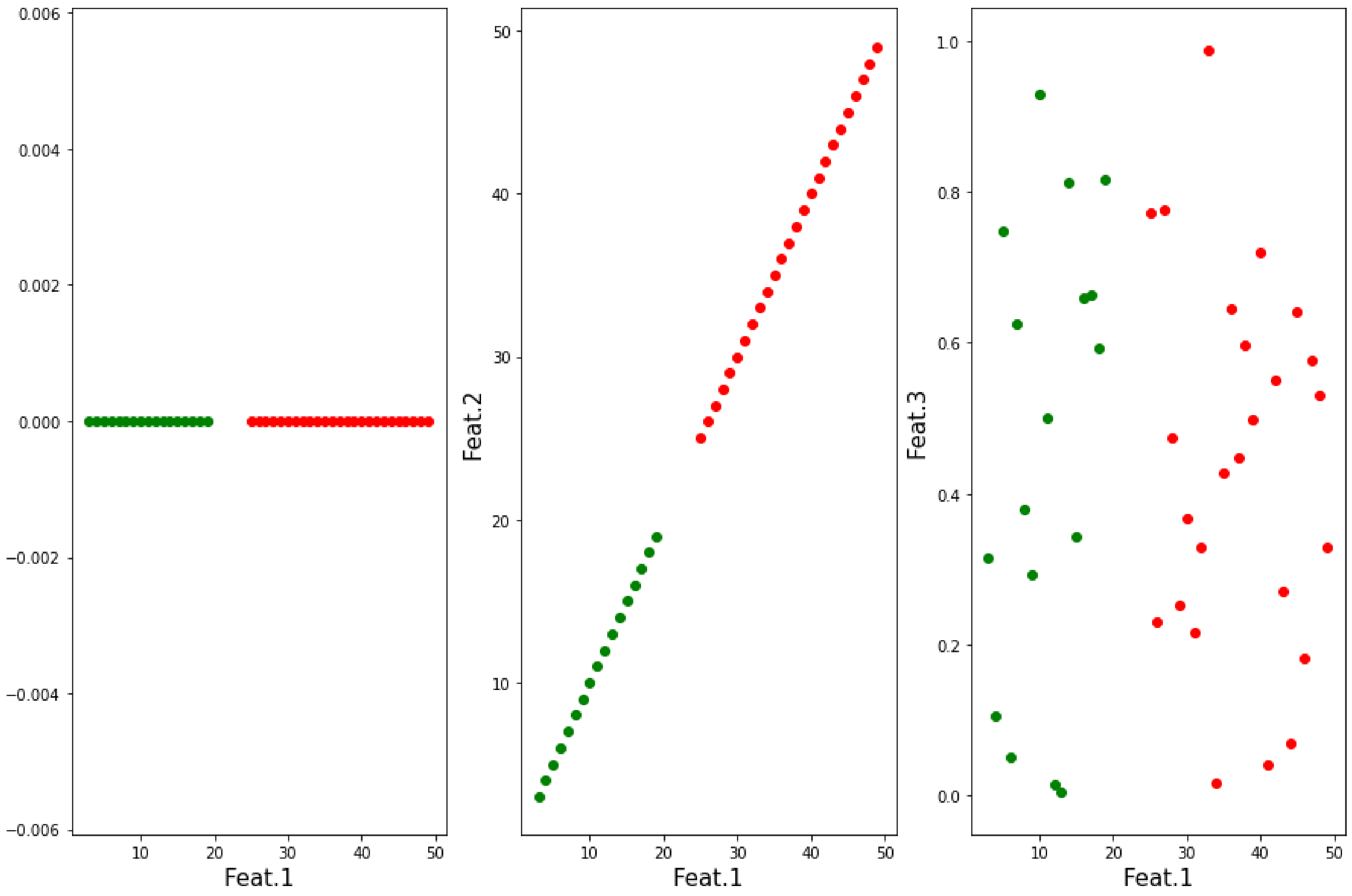
Or, alternatively, you just reduced the dimension of your data from 1 (axis) to 0 (just a point).

Why is this important?

- Visualization: high dimensional data are much harder to visualize; in this case, we only had one variable. But we could have thousands (text data).
- ML model complexity: by having fewer features we can reduce the complexity of our ML models.
- Noise filtering: we only take the most important information.

Imagine you have the following plots

In [126]:
```
plot_feat()
```



And I asked you, to which feature does it statement belong?

- Which is the most relevant feature? Which feature gives us the most information about our data?
- Which is the redundant feature? We could get information from it, but there are easier ways to get it.
- Which is the irrelevant feature?

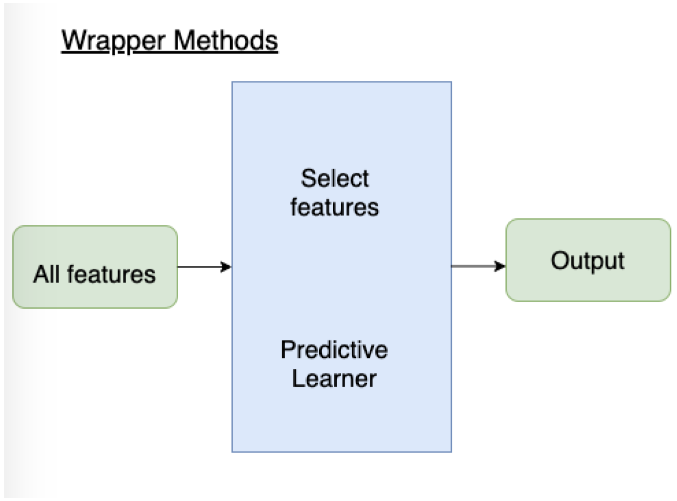
How do we do it?

- Dropping columns (feature selection?)
    - Filter Methods
    - Wrapper Methods
- Creating new features by combinating the given features.
    - Dimentionality Reduction Techniques (such as PCA)

Feature Selection : Wrapper Methods

Learner- dependent method.

We have to iterarate through all our features. This is impractical if we have a large dataset and very complicated!



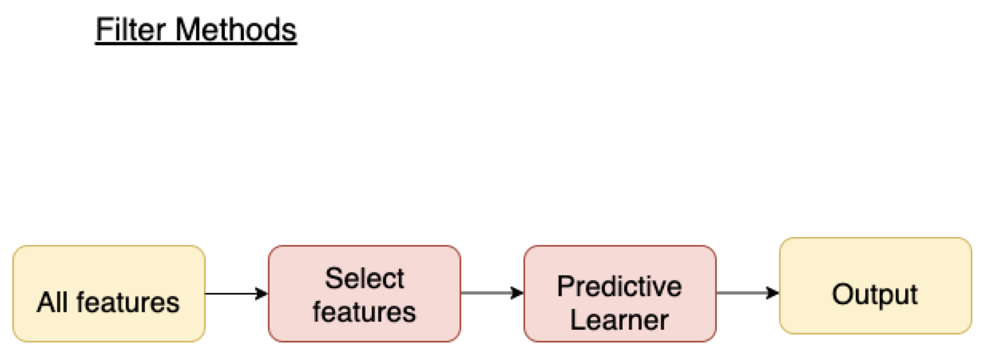
Examples:

- Forward Selection
- Backward Selection
- Stepwise Selection

Feature Selection : Filter Methods

Learner independent method.

Rely on data characteristics



How it works?

- rank feature importance
- filter out less important features
- choose k features
- give new features to the learner and start classifying.

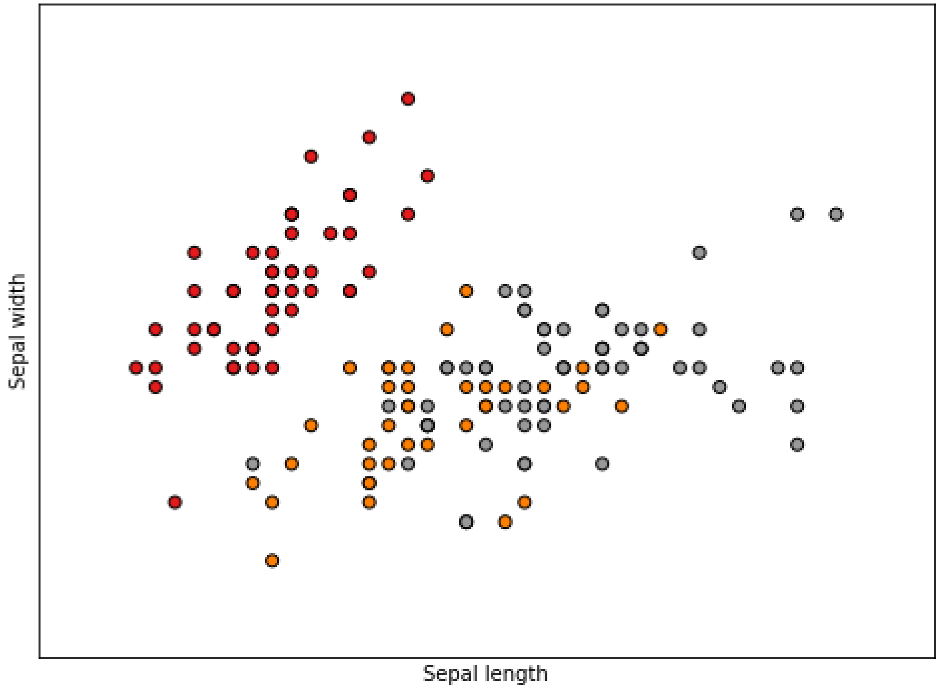
Other Dimentionality Reduction Techniques

What if instead of choosing just some features, we created new features?

Flowers Example

- Suppose you have the sepal length and sepal width of 150 specimens of flowers;

In [128]:
```
plot_iris()
```

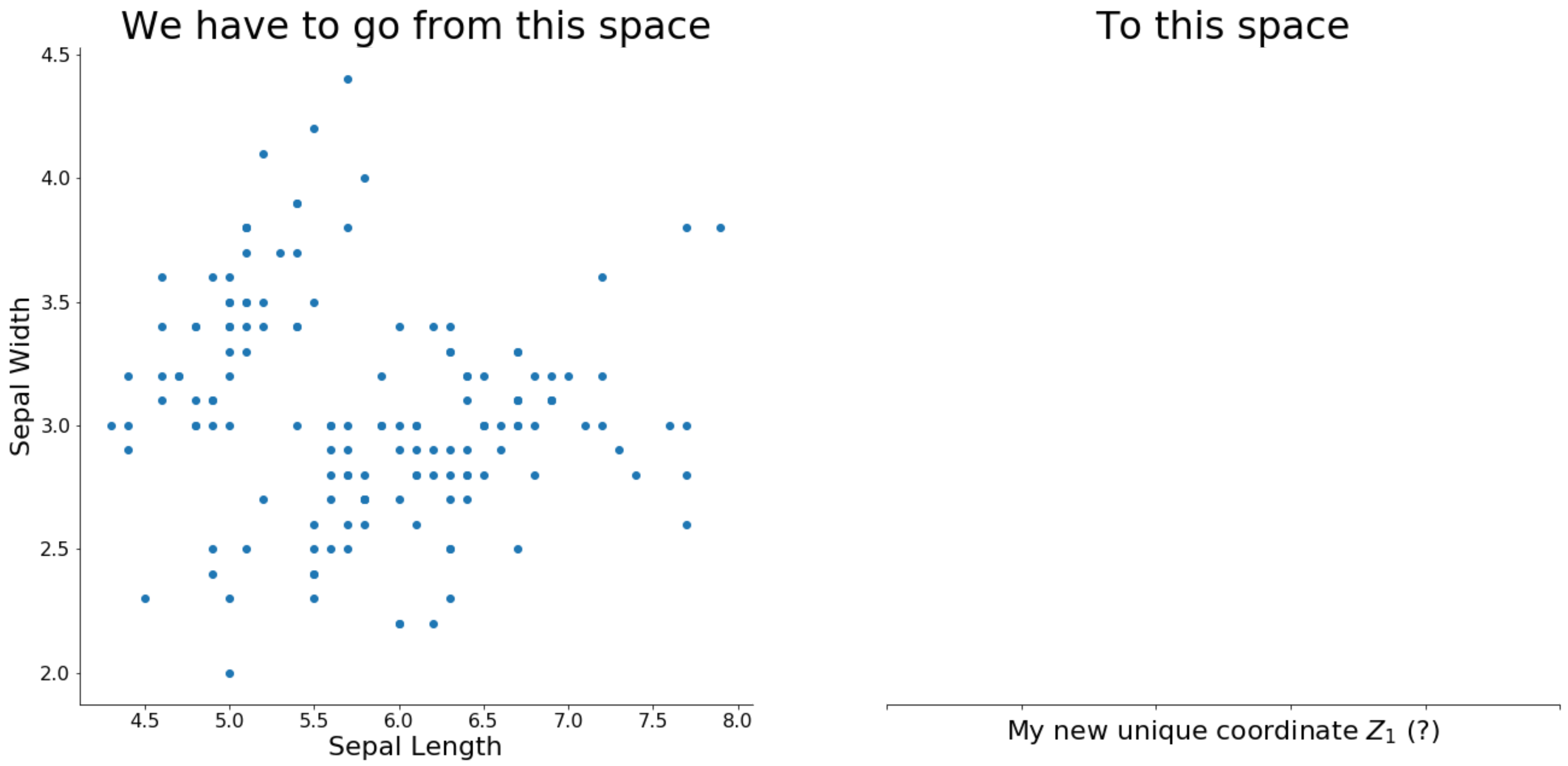


Ok, my computer does not have enough memory for all these data points;

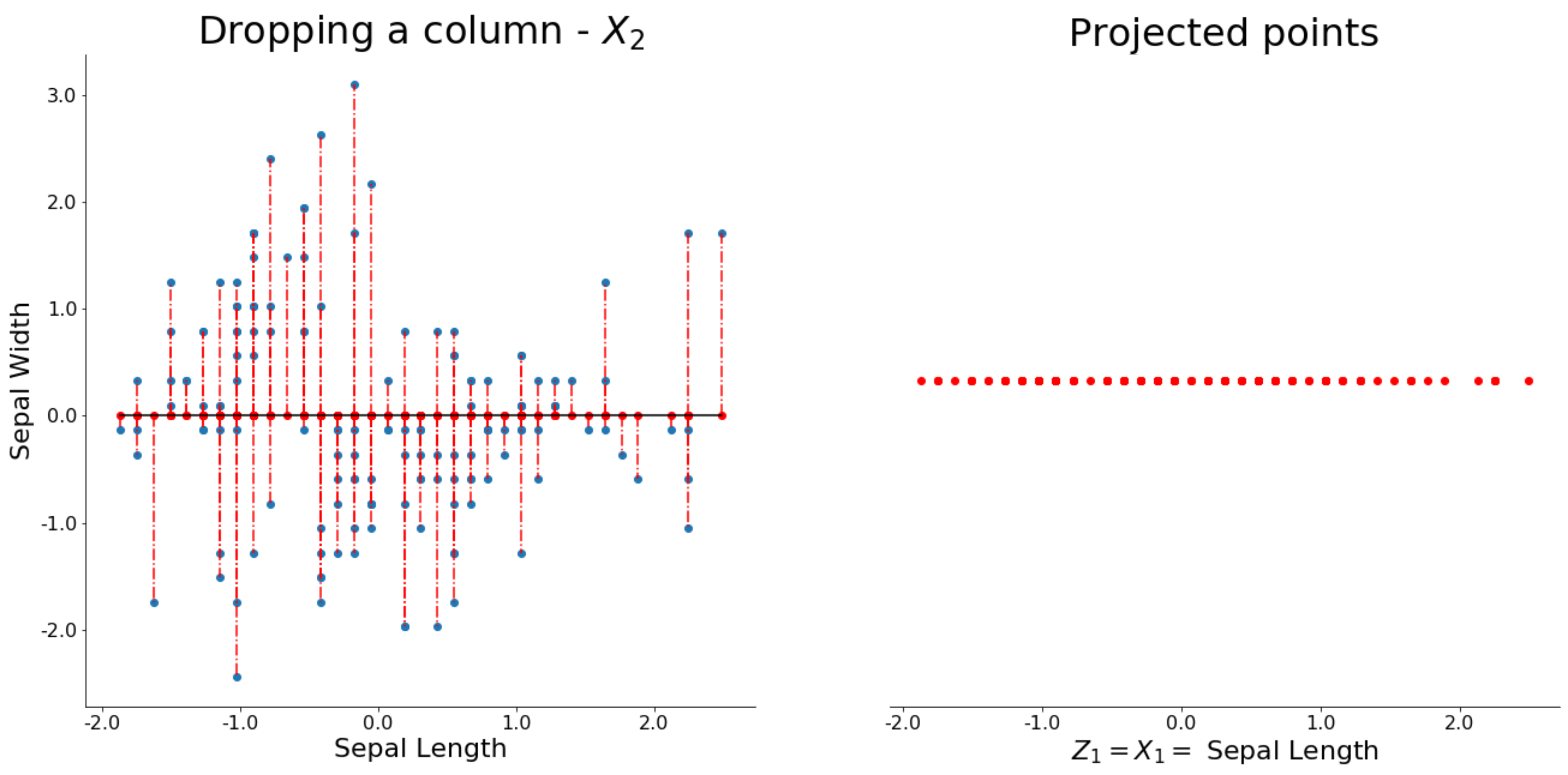
I can only store one coordinate for each observation;

In other words, we get to keep all the rows, but we have to reduce our two-columns matrix to a one-column matrix;

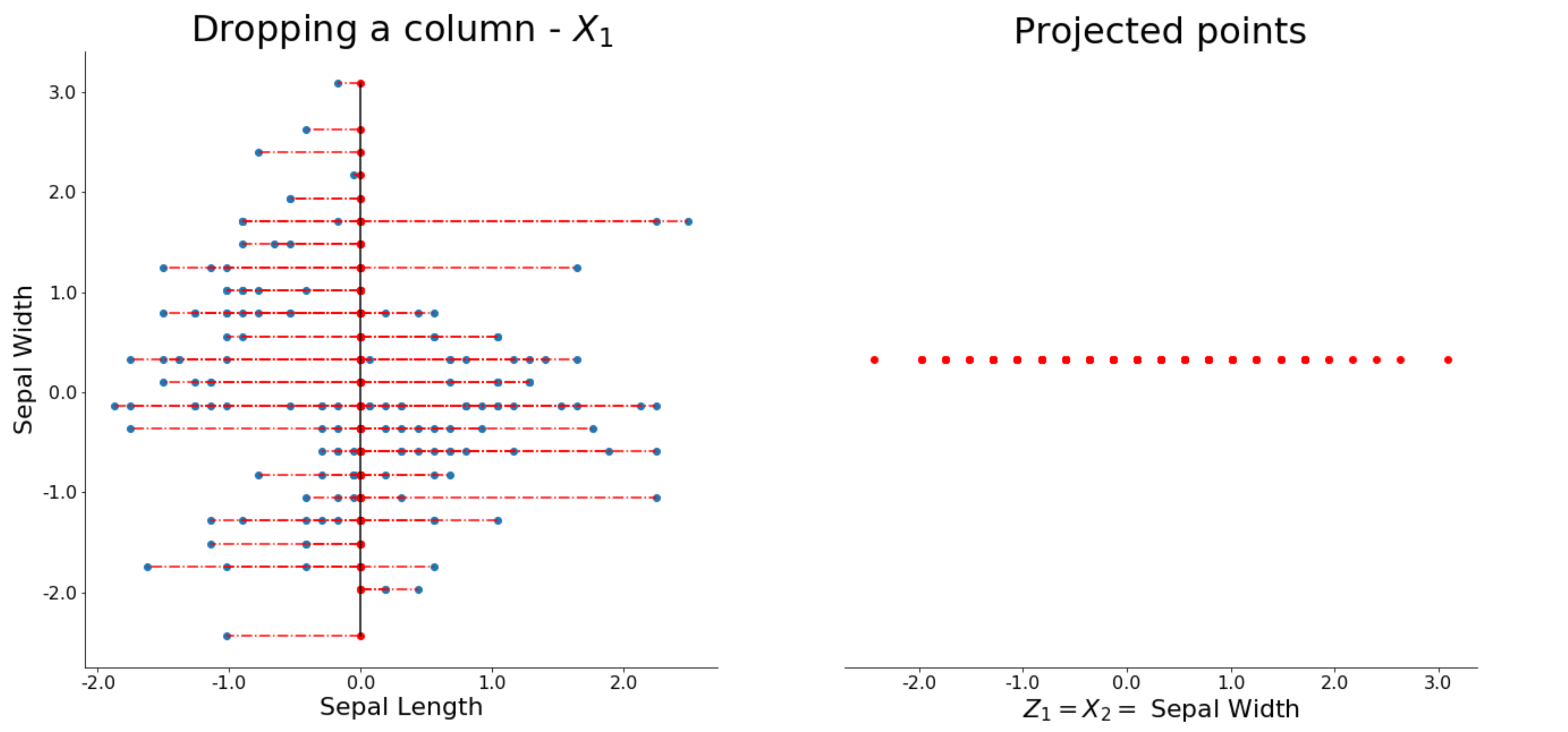
In [130]:
```
plot_iris_reduction2d_1d(22,10)
```



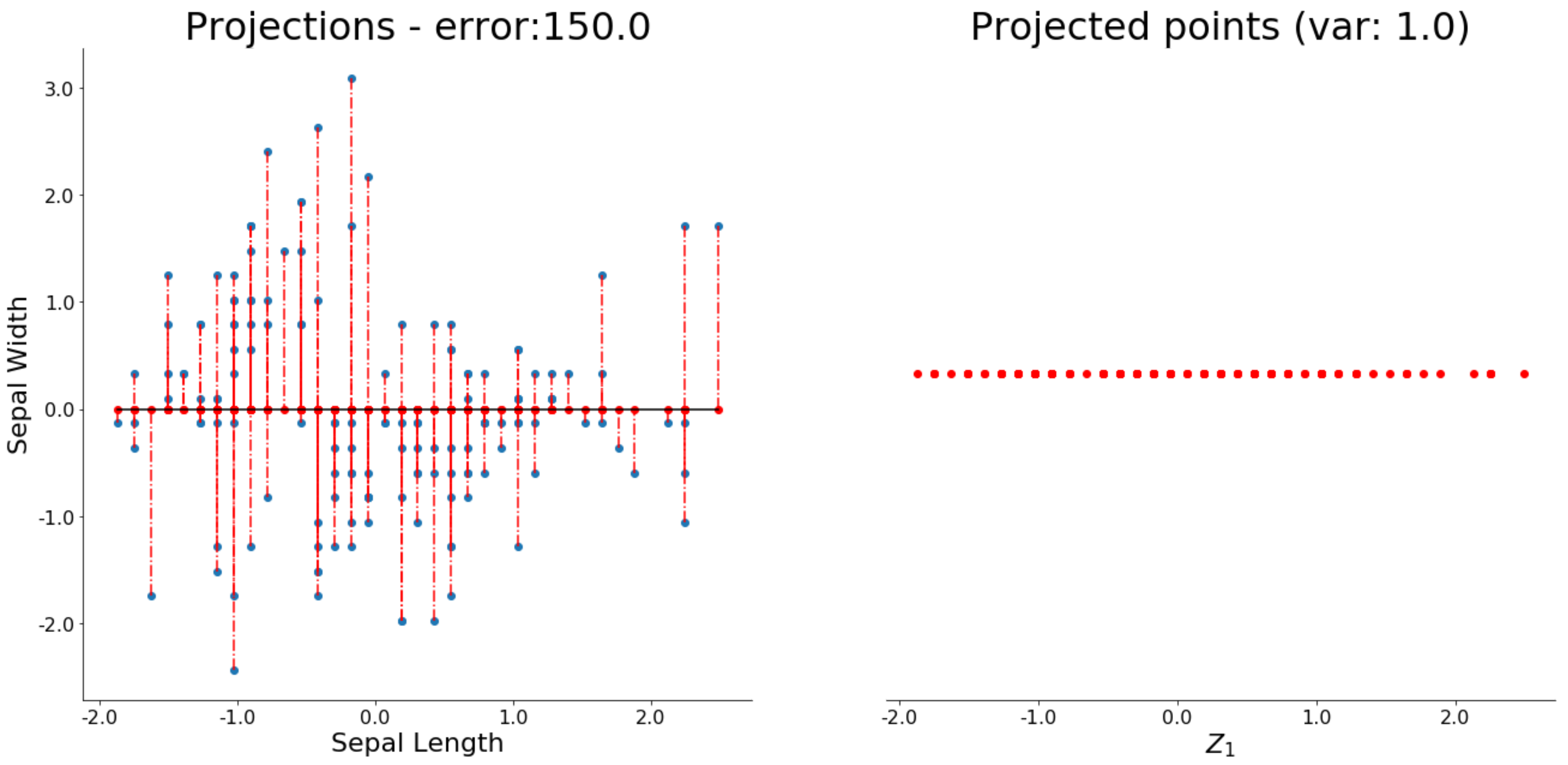
In [133]:
```
plot_iris_projecting_x1(22,10)
```

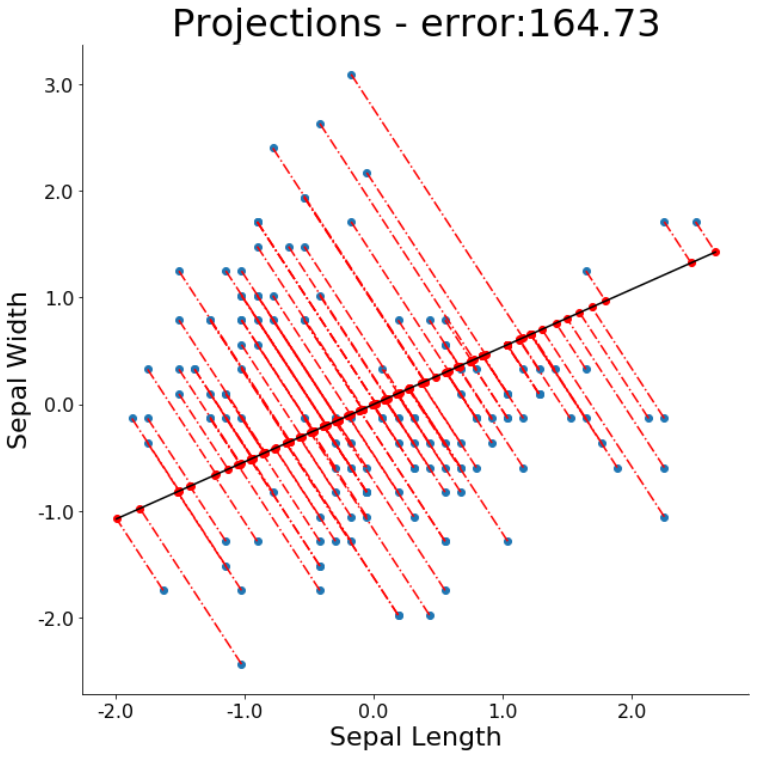


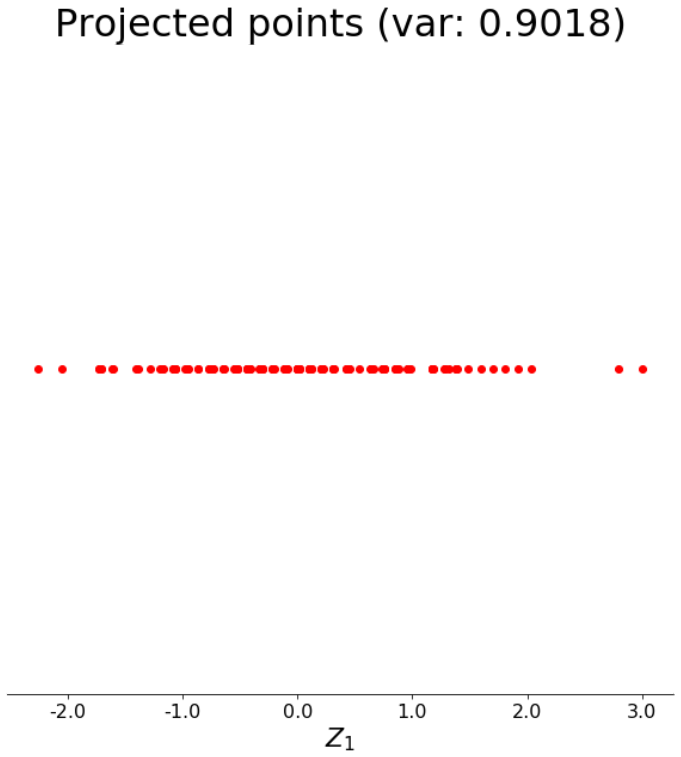
In [134]:
```
plot_iris_projecting_x2(22,10)
```



In [117]:
```
for θ in np.linspace(0,170,7):
    plot_iris_projections(22, 10, θ, clear = False)
    time.sleep(0)
```

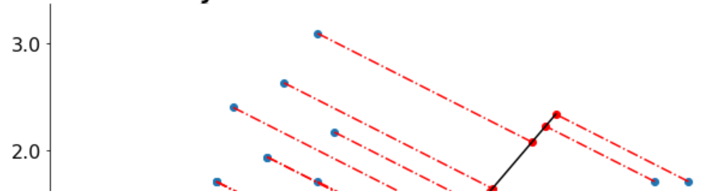



Projections - error:164.73
Sepal Width
Sepal Length
3.0
2.0
1.0
0.0
-1.0
-2.0
-2.0
-1.0
0.0
1.0
2.0


Projected points (var: 0.9018)
Z_1
-2.0
-1.0
0.0
1.0
2.0
3.0

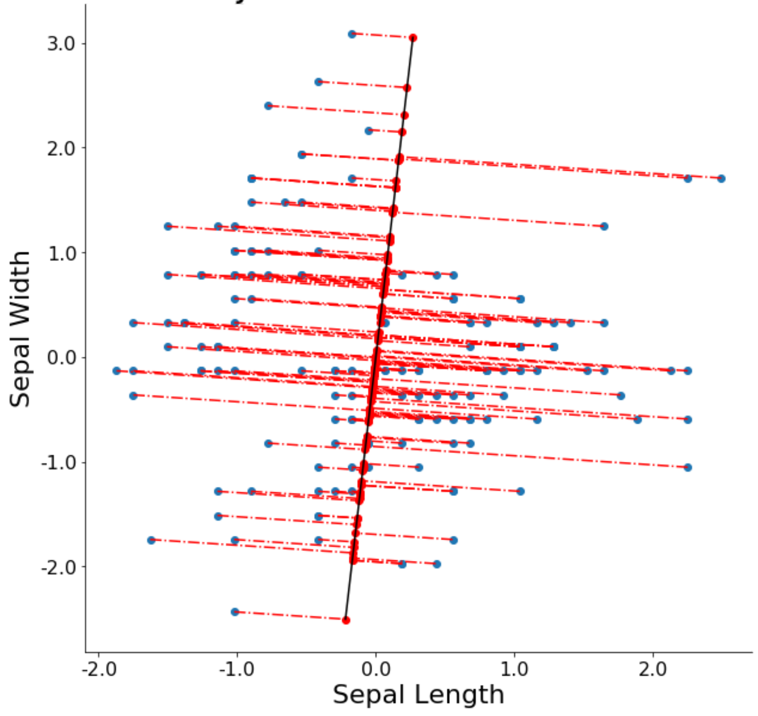
Projections - error:166.19

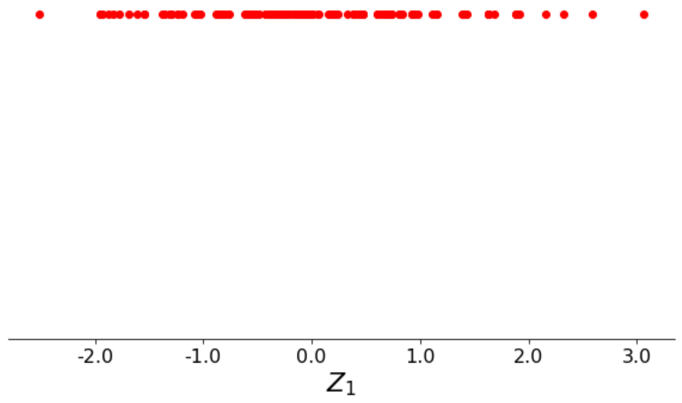
Projected points (var: 0.892)

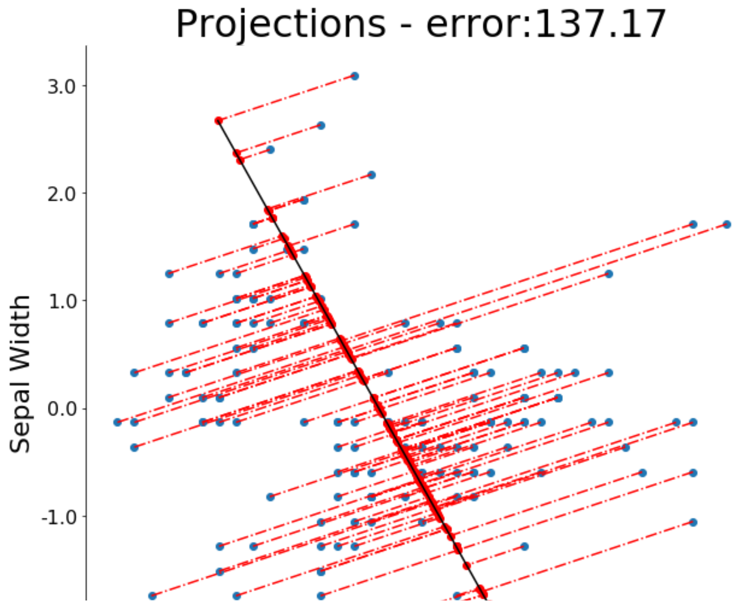


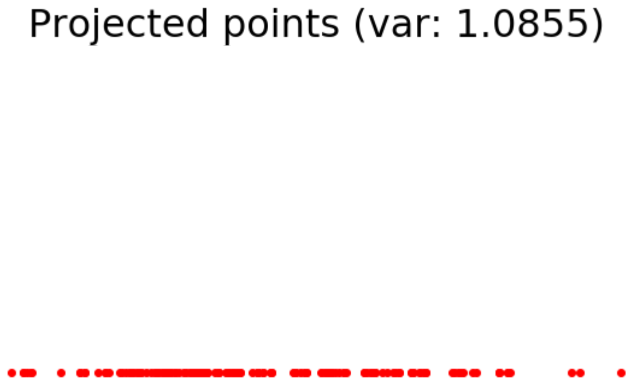
Projections - error:153.06

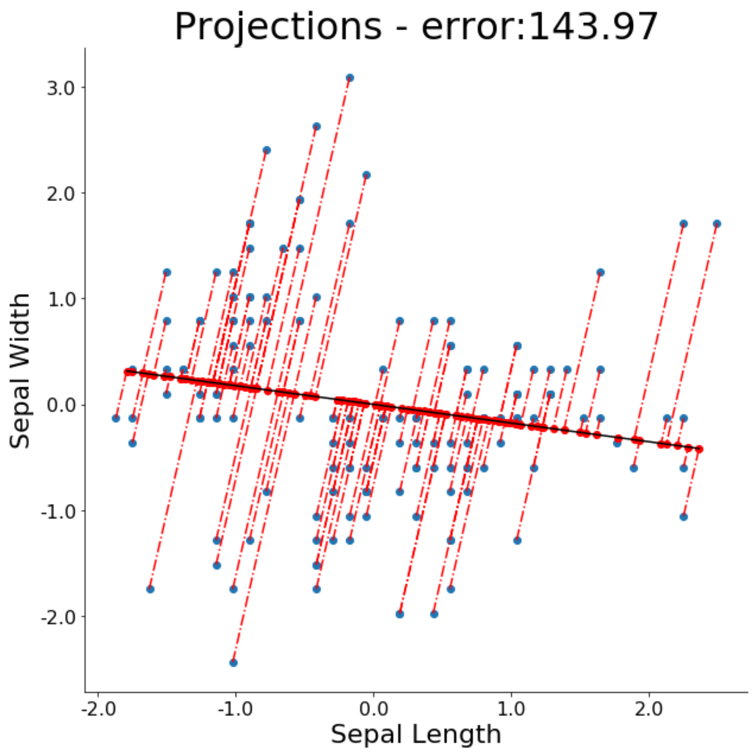
Projected points (var: 0.9796)

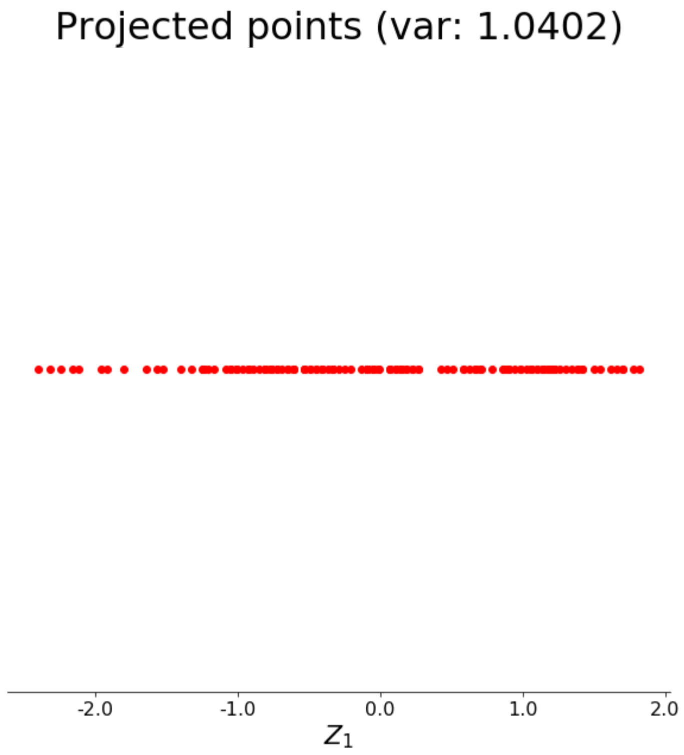




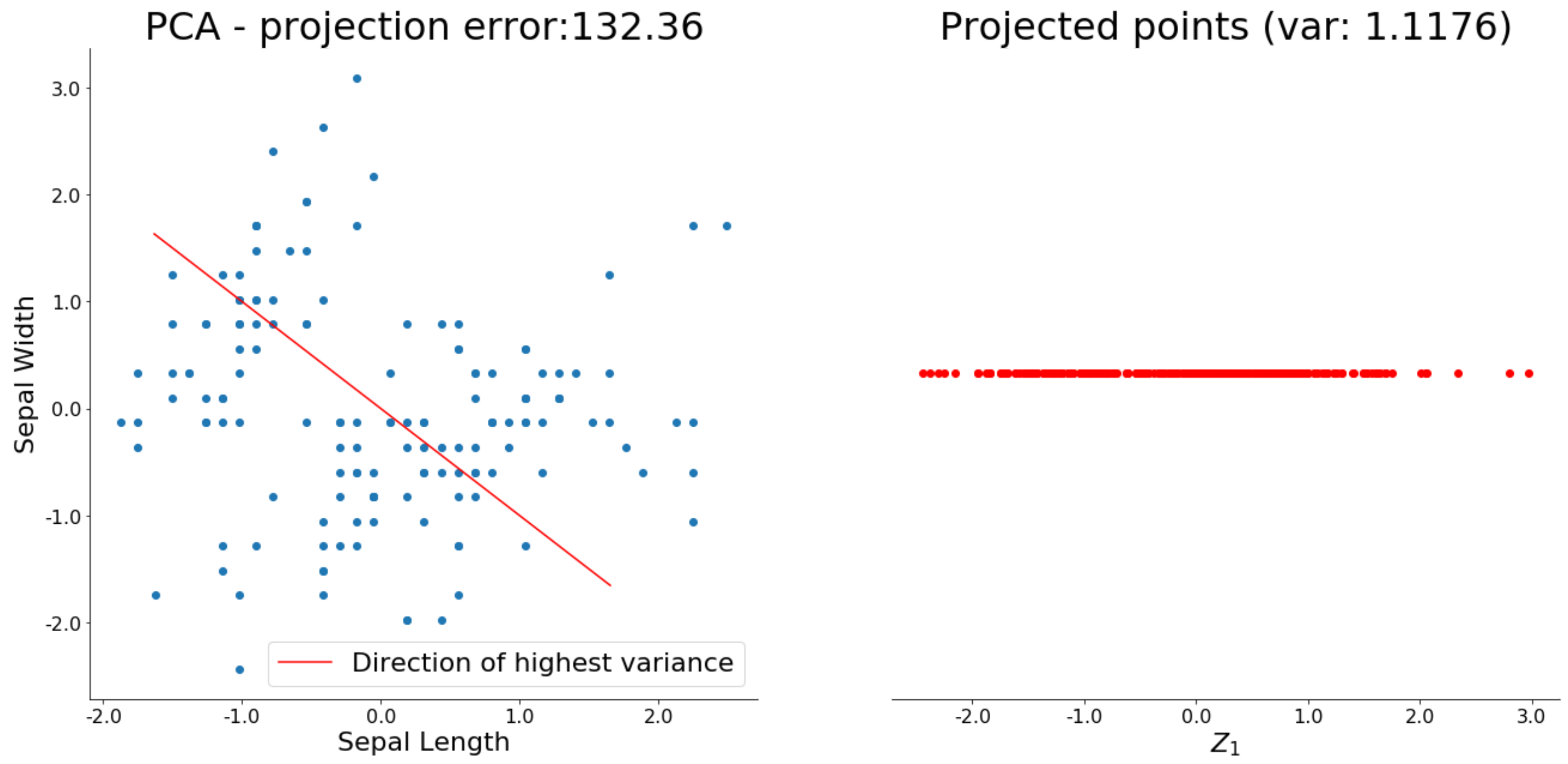

Projections - error:137.17
Sepal Width
3.0
2.0
1.0
0.0
-1.0


Projected points (var: 1.0855)

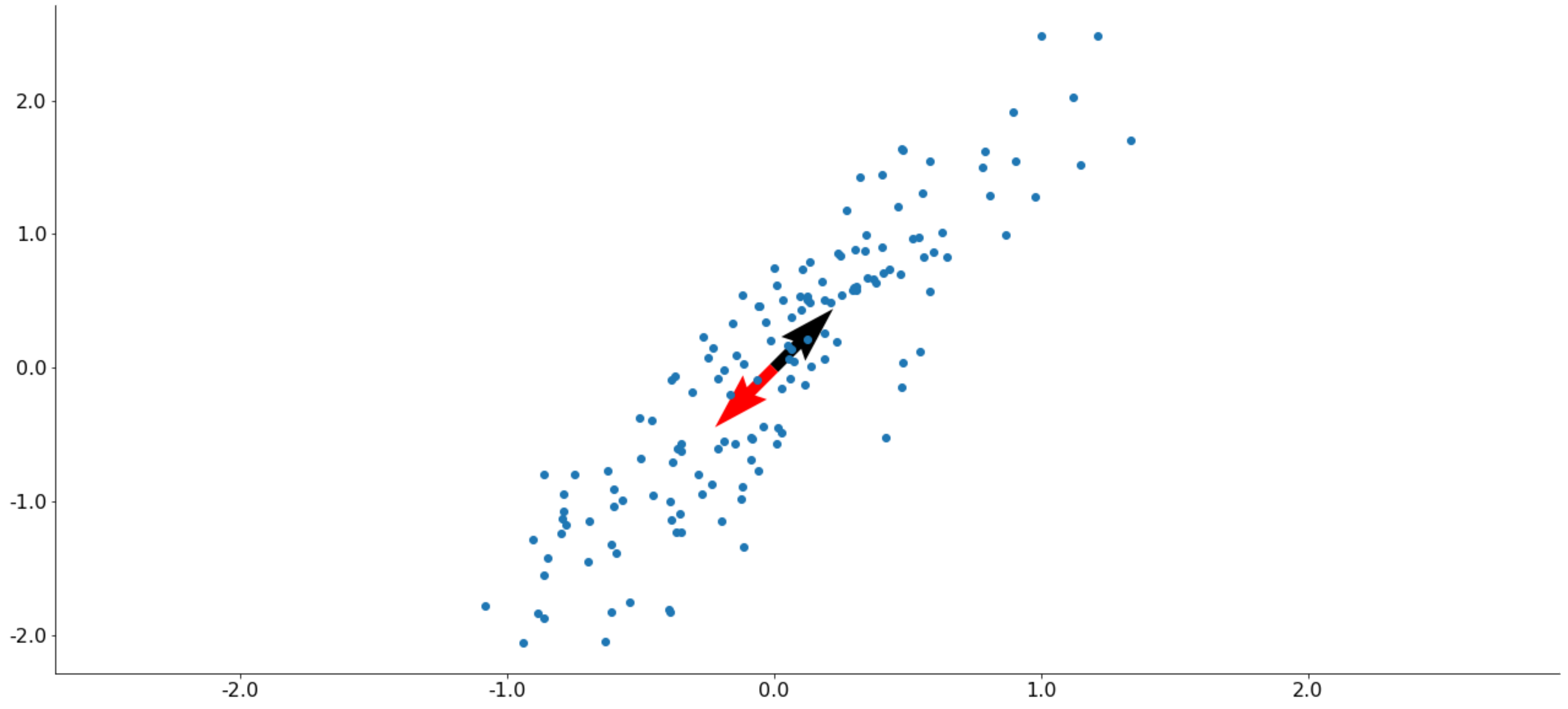

Projections - error:143.97
Sepal Width
Sepal Length
3.0
2.0
1.0
0.0
-1.0
-2.0
-2.0
-1.0
0.0
1.0
2.0


Projected points (var: 1.0402)
Z_1
-2.0
-1.0
0.0
1.0
2.0

```
In [119]: ax = plot_pca_2d(22,10, x_std, errors=False)
          ax[0].set_xlabel("Sepal Length", fontdict={'fontsize':22}) # Changes the label of
           x-axis
          ax[0].set_ylabel("Sepal Width", fontdict={'fontsize':22}); # Changes the label of
           y-axis
```



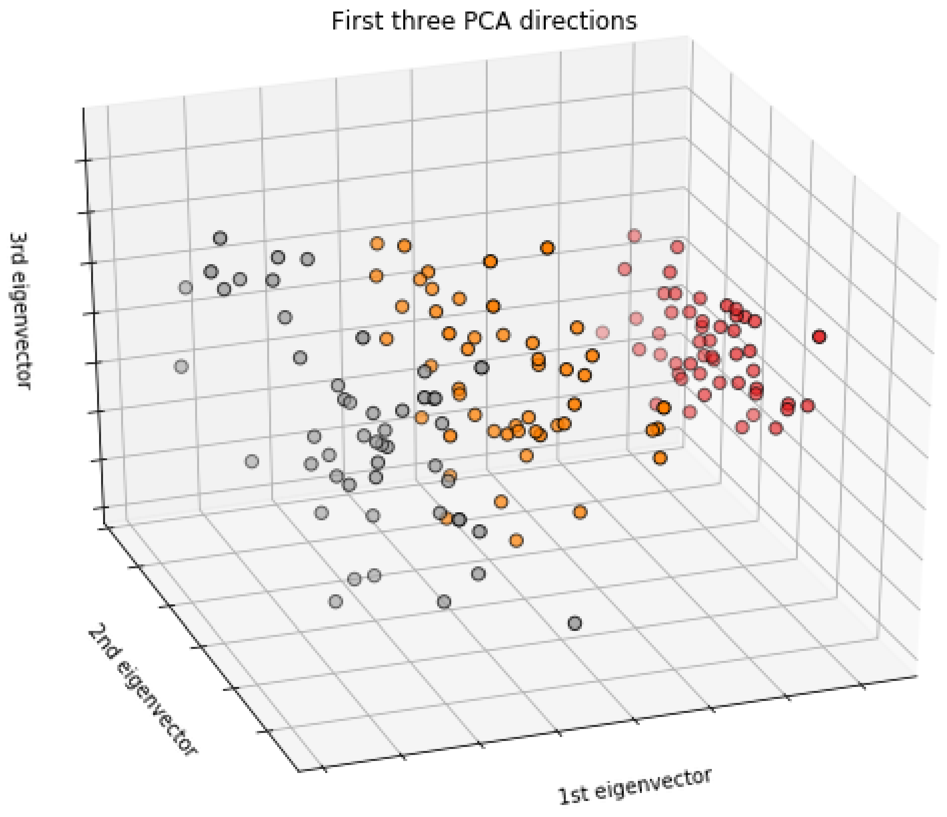
In [143]:
```
plot_pca_mult_solutions(22,10,data_random)
```



We can say that:

- The first principal component is the one with maximum variance;
- The jth principal component is the one that maximizes the variance constrained to be orthogonal to all previously defined components;

```
In [123]: PCA_plot()
```



If you want to learn more about Dimentionality Reduction, come and talk in Mentorship Hours to me!

Thank you!