CSE – 6363 – Machine Learning

Parth Vijaykumar Soni – 1002053647

Final Project Report

Project Description – Detection of volcanoes on Venus using KNN with the help of Principle component analysis.

Overview of the project data – The data was collected by the Magellan spacecraft over an approximately four-year period from 1990--1994. The objective of the mission was to obtain global mapping of the surface of Venus using synthetic aperture radar (SAR). This format consists of two files, a binary file with extension .sdt (the image data) and an ascii file with extension .spr (header information). The .lxyr files are simple space-separated ascii containing label, x-location of centre, y-location of centre, and radius.

Question 1: What your project was about?

Answer 1: The project was to detect volcano on the surface of the Venus as accurate as possible. The project has 4 labels which goes as 1 = definitely a volcano, 2 = probably a volcano, 3 = possibly a volcano, and 4 = only a pit is visible. The images are 1024X1024 pixels. The pixel values are in the range [0,255]. The pixel value is related to the amount of energy backscattered to the radar from a given spatial location. Higher pixel values indicate greater backscatter. Lower pixel values indicate lesser backscatter. Both topography and surface roughness relative to the radar wavelength affect the amount of backscatter. In this project I use KNN along with PCA and I tried to vary the weights which will associate with probability of KNN to predict as the class is imbalanced. The end goal of the project was to correctly identify if a test data fall into any of the above 4 classes.

Question 2: How you went addressing the problem?

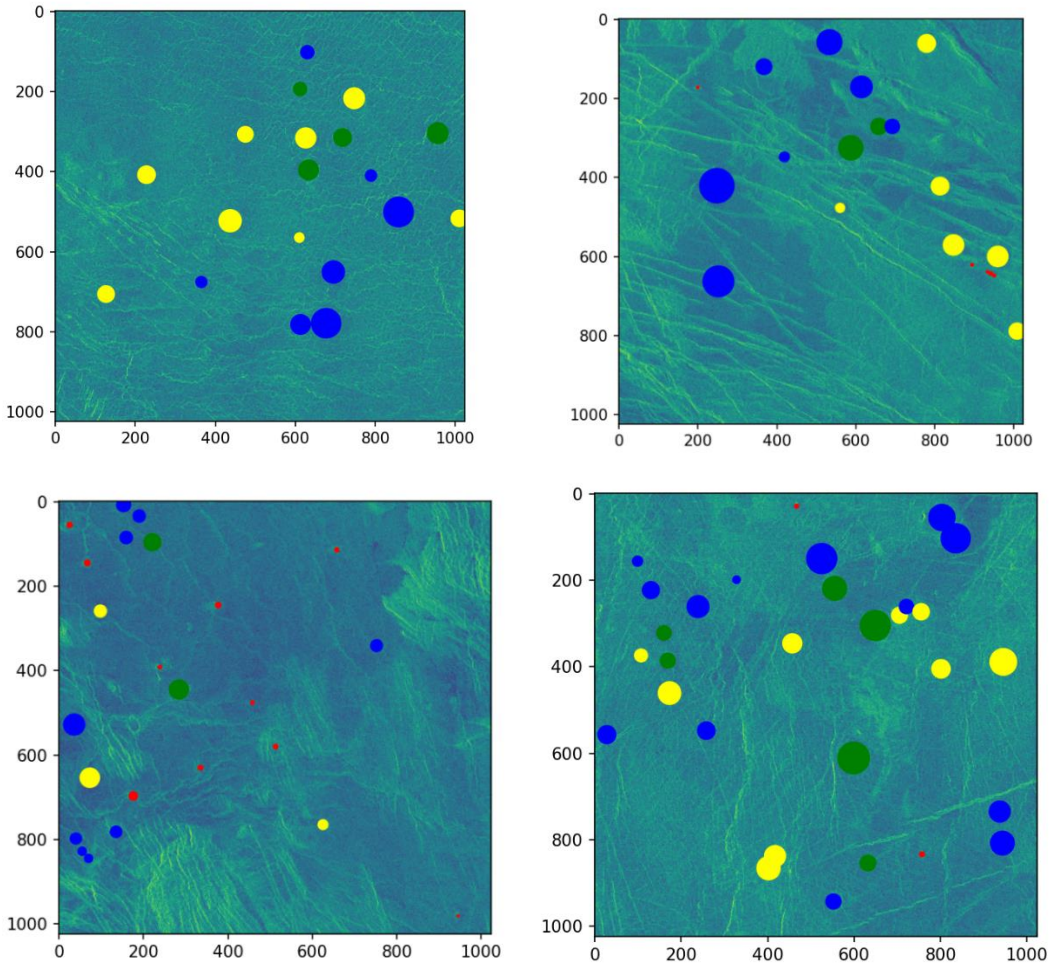
Answer 2: Here are the abstract of the steps which I've followed:

1) Load the image .sdt and .lxyr data.

2) Find out the patches and crop them.
3) Resize the crop image and append ravel image and label to X and Y respectively.
4) Apply PCA to the data.
5) Split the data into train and test.
6) Apply variation of KNN, the variation of KNN as follows:
    a. Update probability consider distance as a weight.
    b. Update probability consider class as a weight.
    c. Update probability consider distance and class as a weight.
    d. Applying bagging.

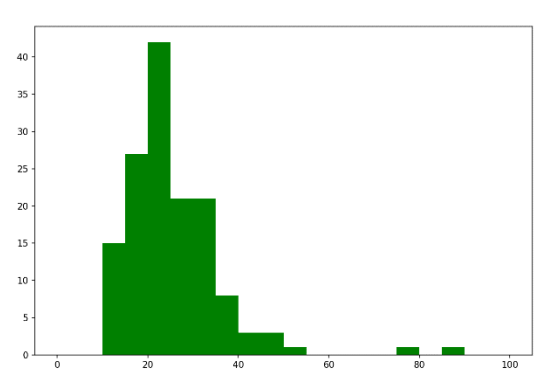
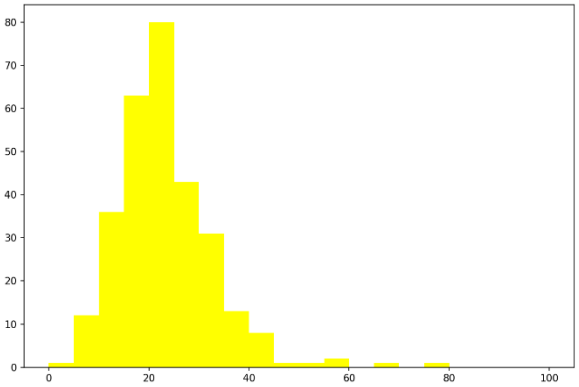
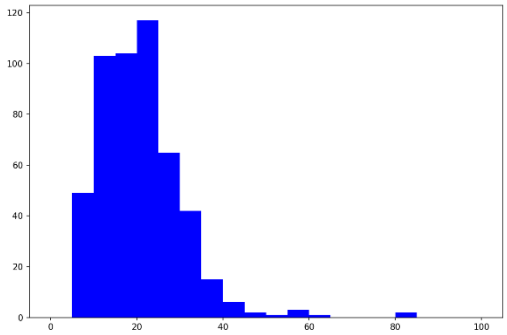
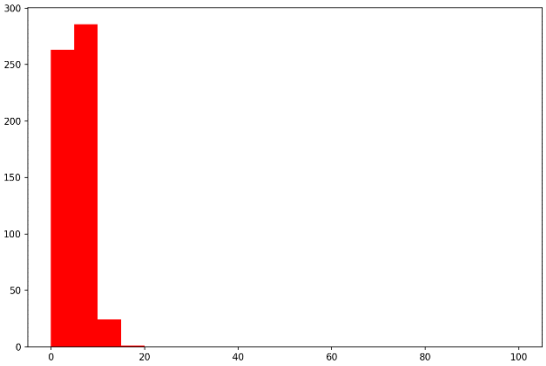
Question 3: How you implemented your solution?

Answer 3: A single image contains more than one patches.



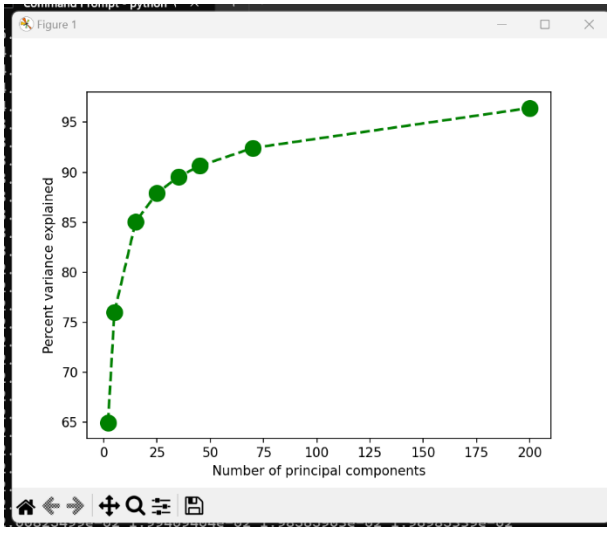
Here red indicates label 4, blue indicates label 3, yellow indicates label 2, and green label 1.

So I cropped the Image based on the x_centre, y_centre, radius. This is the distribution of radius.

So, most of the distribution of radius falls inside 40 unit so I resized it to 40, 40 using sklearn.

And the majority of the variation is covered by the 200 number of components of PCA.



The variation in the data change exponentially from 10 components to 100 components So I will perform a grid for various values of pca that lies in this margin for various values of k.

These are the functions that I have implemented for computing the solution.

1) Data processing – these functions help in gathering X_train and Y_train data by cropping the images of patches and resizing them.
2) PCA – It is the second main function of this program. It helps in finding the first n components of the data which is later used in knn for training and finding out the label for input point
3) Data split – This function helps in splitting data in test and train it consists of two part first part which divides the data randomly on based of split ratio and the second function which divides data such as the number of classes in test data is same for all labels.
4) Knn classifier – it is implemented for finding label for input point and has 3 variation, bagging, using distance as a voting criteria, using class number as voting criteria.
5) Main – in this function I have implemented gird for various values of pca associated with different values of k.
6) The rest of the functions are the helper function which are in a direct or indirect way associated with the above functions.

These all functions along with main and all helper functions are in process.py file and to visualize I have implemented another visualize.py file.

Question 4: What result did you obtain?

Answer 4: Using various values of PCA and k

```
pca_components = [30, 40, 50, 60, 70, 80, 100, 120, 140, 160]
k_values = [5, 7, 9, 11, 13, 15]
```

Results obtained.

```
Accuracy results:
PCA     k=5     k=7     k=9     k=11    k=13    k=15
30      65.263  65.263  65.000  63.684  65.263  64.474
40      65.000  67.105  66.316  66.579  66.316  66.842
50      65.789  66.316  65.789  66.053  66.053  65.789
60      63.158  65.789  65.789  65.789  65.526  66.316
70      64.211  65.000  67.632  65.263  65.000  66.053
80      63.684  66.316  66.316  65.789  66.316  65.789
100     63.421  66.316  65.789  64.737  65.789  65.789
120     63.421  65.263  66.316  66.579  66.053  66.316
140     64.211  65.526  66.316  66.053  66.842  66.842
160     65.000  65.789  66.579  66.842  66.579  67.368
```

Accuracy results using distance as a weight for calculating probability in knn:

PCA	k=5	k=7	k=9	k=11	k=13	k=15
30	65.263	65.263	65.263	63.684	65.526	64.211
40	65.000	66.579	67.105	66.579	66.579	66.316
50	66.053	65.789	67.105	66.053	66.053	65.263
60	63.158	65.789	66.579	66.316	65.526	65.789
70	64.474	65.263	68.421	65.789	65.263	65.789
80	64.211	65.789	66.579	65.789	66.316	65.789
100	62.632	65.789	65.789	64.737	66.053	65.526
120	63.158	64.737	66.579	66.053	66.053	66.316
140	63.947	64.737	66.316	66.316	66.842	66.579
160	63.684	64.737	66.053	66.579	66.053	66.842

Accuracy results using number of class as a weight for calculating probability in knn:

PCA	k=5	k=7	k=9	k=11	k=13	k=15
30	54.211	60.263	56.842	56.053	59.737	60.526
40	54.737	60.526	59.737	58.158	59.211	60.526
50	56.842	61.316	58.684	59.211	61.579	60.789
60	56.053	61.842	60.526	58.947	61.316	61.053
70	57.105	61.053	61.053	61.579	60.526	60.526
80	56.053	62.368	61.053	61.579	60.263	60.789
100	56.316	61.579	61.579	61.579	58.947	61.579
120	56.579	62.368	60.263	59.737	58.684	60.789
140	58.421	60.789	60.000	59.737	59.474	60.000
160	58.158	60.263	57.105	58.684	57.895	59.474

The above tables are the pictures of the output which is shown as a tabular format. The above uses pca which can explain data variation by more than 85%.

```
pca_components = [2, 3, 5, 7, 10]
k_values = [1, 2, 3, 5, 7]
```

Accuracy results:

PCA	k=1	k=2	k=3	k=5	k=7
2	47.895	47.895	52.632	55.000	55.526
3	57.105	57.105	60.526	61.053	64.737
5	60.000	60.000	60.000	63.684	62.632
7	57.105	57.105	58.684	61.053	64.474
10	61.316	61.316	63.684	64.211	63.158

Accuracy results using distance as a weight for calculating probability in knn:

PCA	k=1	k=2	k=3	k=5	k=7
2	47.895	47.895	50.526	52.368	52.632
3	57.105	57.105	60.526	60.263	62.895
5	60.000	60.000	60.263	63.684	62.368
7	57.105	57.105	58.421	61.842	64.737
10	61.316	61.316	63.684	64.474	63.421

Accuracy results using number of class as a weight for calculating probability in knn:

PCA	k=1	k=2	k=3	k=5	k=7
2	47.895	46.842	49.737	49.474	50.263
3	57.105	52.632	52.632	50.526	55.000
5	60.000	52.632	54.737	51.842	56.053
7	57.105	53.947	56.053	56.053	56.053
10	61.316	54.474	58.684	55.000	58.421

For checking the accuracy class wise for test data I have implemented another part of grid for that.

The result obtained for values or PCA and k are

```
for pca in [2,5,10,15,20,30,60,90,120,150,200]:
    print(f"Applying for PCA = {pca}")
    accuracy_results_entire_k = []
    k_values = [2,3,5,7,9,11,13,15]
```

```
Applying for PCA = 2
K       Class 1.0       Class 2.0       Class 3.0       Class 4.0
2       0.179           0.435           0.306           0.725
3       0.308           0.290           0.373           0.775
5       0.410           0.333           0.313           0.775
7       0.231           0.362           0.410           0.739
9       0.282           0.377           0.343           0.732
11      0.333           0.406           0.313           0.754
13      0.333           0.435           0.306           0.746
15      0.282           0.348           0.313           0.739
```

```
Applying for PCA = 5
K       Class 1.0       Class 2.0       Class 3.0       Class 4
2       0.231           0.435           0.381           0.797
3       0.333           0.232           0.463           0.848
5       0.436           0.406           0.284           0.826
7       0.256           0.391           0.463           0.826
9       0.333           0.362           0.425           0.862
11      0.333           0.391           0.396           0.862
13      0.333           0.348           0.396           0.862
15      0.333           0.377           0.425           0.855
```

```
Applying for PCA = 10
K       Class 1.0       Class 2.0       Class 3.0       Class 4.0
2       0.308           0.478           0.358           0.826
3       0.359           0.188           0.560           0.877
5       0.538           0.333           0.351           0.855
7       0.385           0.261           0.560           0.826
9       0.436           0.348           0.425           0.841
11      0.487           0.261           0.448           0.841
13      0.436           0.304           0.500           0.841
15      0.487           0.304           0.515           0.833
```

Applying for PCA = 15

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.410	0.464	0.425	0.833
3	0.590	0.159	0.590	0.870
5	0.564	0.449	0.403	0.812
7	0.410	0.304	0.530	0.797
9	0.462	0.290	0.433	0.826
11	0.538	0.246	0.507	0.841
13	0.462	0.304	0.515	0.812
15	0.462	0.319	0.522	0.812

Applying for PCA = 20

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.513	0.507	0.388	0.797
3	0.564	0.203	0.537	0.855
5	0.641	0.333	0.366	0.841
7	0.385	0.319	0.545	0.812
9	0.564	0.319	0.433	0.819
11	0.513	0.203	0.485	0.833
13	0.487	0.290	0.545	0.826
15	0.513	0.275	0.560	0.812

Applying for PCA = 30

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.385	0.478	0.396	0.797
3	0.564	0.188	0.560	0.848
5	0.538	0.319	0.366	0.826
7	0.385	0.319	0.582	0.826
9	0.513	0.333	0.433	0.833
11	0.538	0.275	0.455	0.812
13	0.564	0.290	0.545	0.812
15	0.564	0.261	0.590	0.804

Applying for PCA = 60

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.410	0.449	0.433	0.797
3	0.615	0.174	0.552	0.855
5	0.590	0.333	0.388	0.833
7	0.513	0.377	0.560	0.826
9	0.564	0.348	0.507	0.841
11	0.513	0.304	0.515	0.826
13	0.564	0.333	0.552	0.826
15	0.538	0.348	0.575	0.797

Applying for PCA = 90

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.462	0.478	0.418	0.819
3	0.564	0.203	0.530	0.855
5	0.590	0.348	0.403	0.848
7	0.538	0.333	0.545	0.833
9	0.590	0.420	0.515	0.848
11	0.538	0.377	0.552	0.833
13	0.487	0.290	0.545	0.841
15	0.564	0.290	0.552	0.826

Applying for PCA = 120

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.513	0.493	0.388	0.833
3	0.538	0.188	0.530	0.877
5	0.487	0.362	0.396	0.855
7	0.487	0.377	0.567	0.841
9	0.487	0.377	0.493	0.855
11	0.462	0.290	0.545	0.841
13	0.436	0.261	0.530	0.848
15	0.462	0.348	0.545	0.841

Applying for PCA = 150

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.513	0.507	0.410	0.848
3	0.590	0.217	0.515	0.891
5	0.590	0.377	0.410	0.855
7	0.487	0.304	0.575	0.848
9	0.487	0.304	0.440	0.855
11	0.462	0.275	0.507	0.862
13	0.436	0.319	0.515	0.848
15	0.410	0.319	0.522	0.848

Applying for PCA = 200

K	Class 1.0	Class 2.0	Class 3.0	Class 4.0
2	0.487	0.464	0.418	0.877
3	0.615	0.174	0.485	0.899
5	0.564	0.391	0.433	0.877
7	0.462	0.304	0.522	0.855
9	0.436	0.333	0.463	0.870
11	0.436	0.275	0.530	0.862
13	0.410	0.261	0.507	0.870
15	0.487	0.275	0.545	0.877

These are the results obtained.

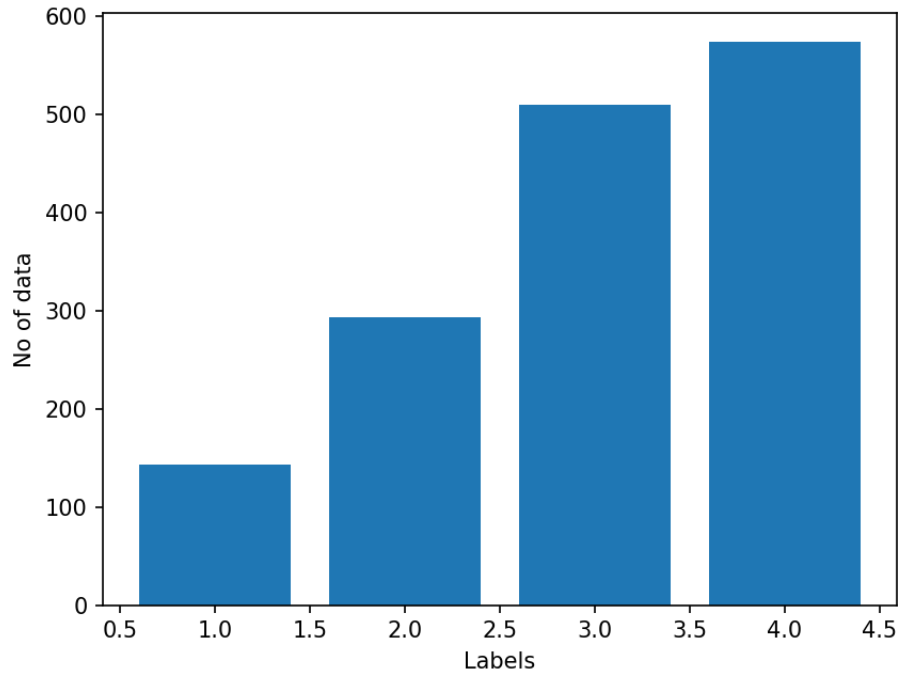
Question – 5: A discussion of what these results show.

Answer – 5: The above results are the results obtained by applying various values for values of pca and k. As it shown in the graph of variation of data explained there is drastic change in between the 2 – 15 number of components so as the above result support that

```
Accuracy results:
PCA     k=1     k=2     k=3     k=5     k=7
2       47.895  47.895  52.632  55.000  55.526
3       57.105  57.105  60.526  61.053  64.737
5       60.000  60.000  60.000  63.684  62.632
7       57.105  57.105  58.684  61.053  64.474
10      61.316  61.316  63.684  64.211  63.158
Accuracy results using distance as a weight for calculating probability in knn:
PCA     k=1     k=2     k=3     k=5     k=7
2       47.895  47.895  50.526  52.368  52.632
3       57.105  57.105  60.526  60.263  62.895
5       60.000  60.000  60.263  63.684  62.368
7       57.105  57.105  58.421  61.842  64.737
10      61.316  61.316  63.684  64.474  63.421
Accuracy results using number of class as a weight for calculating probability in knn:
PCA     k=1     k=2     k=3     k=5     k=7
2       47.895  46.842  49.737  49.474  50.263
3       57.105  52.632  52.632  50.526  55.000
5       60.000  52.632  54.737  51.842  56.053
7       57.105  53.947  56.053  56.053  56.053
10      61.316  54.474  58.684  55.000  58.421
```

This result obtained supports the claim of data variation and so the accuracy is increased between pca 2 to 10. However, there not much change in data variation in between pca 30 to 150 and so the accuracy lies in between 65 – 67 for that and not much change is observed.

Another thing which I observed that the number of only a pit is more than other label.



It covers around 30% of dataset. And, so is explained when I saw the class wise accuracy most of the classifier predict the accuracy of class 4 of more than 75%. Which is caused by imbalanced dataset.