HW 8

This assignment covers all fundamental concepts required for completing a project

DO NOT ERASE MARKDOWN CELLS AND INSTRUCTIONS IN YOUR HW submission

- **Q** - QUESTION
- **A** - Where to input your answer

Instructions

Keep the following in mind for all notebooks you develop:

- Structure your notebook.
- Use headings with meaningful levels in Markdown cells, and explain the questions each piece of code is to answer or the reason it is there.
- Make sure your notebook can always be rerun from top to bottom.
- Please start working on this assignment as soon as possible. If you are a beginner in Python this might take a long time. One of the objectives of this assignment is to help you learn python and scikit-learn package.
- See README.md (README.md) for homework submission instructions

Related Tutorials

Refreshers

- Intro to Machine Learning w scikit-learn (https://scikit-learn.org/stable/tutorial/basic/tutorial.html)
- A tutorial on statistical-learning for scientific data processing (https://scikit-learn.org/stable/tutorial/statistical_inference/index.html#stat-learn-tut-index)

Classification Approaches

- Logistic Regression with Sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
- KNN with sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)
- Support Vector machine example (https://scikit-learn.org/stable/auto_examples/exercises/plot_iris_exercise.html#sphx-glr-auto-examples-exercises-plot-iris-exercise-py)
- SVC (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC)
- Bagging Classifier (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html)
- Gradient Boosting Classifier (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html)

Modeling

- [Cross-validation (https://scikit-learn.org/stable/modules/cross_validation.html)](https://scikit-learn.org/stable/modules/cross_validation.html)
- [Plot Confursion Matrix with Sklearn (https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)](https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)
- [Confusion Matrix Display (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html#sklearn.metrics)](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html#sklearn.metrics)

Import all required library

In [16]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, MaxAbsScaler
import json
import lightgbm as lgbm
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler
```

Data Processing

Q1 Get training data from the dataframe

1. Load `HW8_data.csv` from `data` folder into data frame
2. Print the head of the dataframe
3. Print the shape of the dataframe
4. Print the description of the dataframe
5. Assign `Cover_Type` values to Y
6. Assign rest of the column values to X

A1 Fill the cell blocks below, Create new cell as per your necessary

In [151]:

```
df = pd.read_csv('C:/Users/pedro/Documents/2022 Spring Semester/Machine Le
df.head(10)
```

Out[151]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrolo
0	3080	137	18	166	
1	2758	19	8	551	
2	2779	86	9	43	-
3	2811	296	0	287	

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrolo
4	2956	314	26	71	
5	2638	108	11	396	
6	2956	108	29	291	
7	2625	264	8	396	
8	2751	150	25	263	
9	2947	83	22	662	

In [152]:

Out[152]: (80000, 55)

In [153]:

Out[153]:

	count	mean	std	min	25%	50
Elevation	80000.0	2981.434325	287.972764	1813.0	2762.0	2967.
Aspect	80000.0	151.634175	109.945631	-29.0	60.0	122.
Slope	80000.0	15.093913	8.531364	-3.0	9.0	14.
Horizontal_Distance_To_Hydrology	80000.0	271.564212	227.532197	-43.0	111.0	212.
Vertical_Distance_To_Hydrology	80000.0	51.510737	68.091489	-276.0	4.0	31.
Horizontal_Distance_To_Roadways	80000.0	1770.080712	1318.661060	-238.0	821.0	1440.
Hillshade_9am	80000.0	211.781612	30.814815	10.0	198.0	218.
Hillshade_Noon	80000.0	221.069125	22.191030	69.0	210.0	224.
Hillshade_3pm	80000.0	140.711750	43.859689	-48.0	115.0	142.
Horizontal_Distance_To_Fire_Points	80000.0	1577.937550	1126.514346	-218.0	781.0	1361.
Wilderness_Area1	80000.0	0.258813	0.437985	0.0	0.0	0.
Wilderness_Area2	80000.0	0.042425	0.201558	0.0	0.0	0.
Wilderness_Area3	80000.0	0.656800	0.474781	0.0	0.0	1.
Wilderness_Area4	80000.0	0.021462	0.144921	0.0	0.0	0.
Soil_Type1	80000.0	0.016762	0.128381	0.0	0.0	0.
Soil_Type2	80000.0	0.031062	0.173488	0.0	0.0	0.
Soil_Type3	80000.0	0.004213	0.064767	0.0	0.0	0.
Soil_Type4	80000.0	0.037012	0.188794	0.0	0.0	0.
Soil_Type5	80000.0	0.015900	0.125090	0.0	0.0	0.
Soil_Type6	80000.0	0.007363	0.085489	0.0	0.0	0.
Soil_Type7	80000.0	0.000000	0.000000	0.0	0.0	0.
Soil_Type8	80000.0	0.002938	0.054119	0.0	0.0	0.
Soil_Type9	80000.0	0.010663	0.102708	0.0	0.0	0.
Soil_Type10	80000.0	0.054363	0.226733	0.0	0.0	0.

	count	mean	std	min	25%	50%
Soil_Type11	80000.0	0.027787	0.164365	0.0	0.0	0.
Soil_Type12	80000.0	0.018200	0.133675	0.0	0.0	0.
Soil_Type13	80000.0	0.031313	0.174162	0.0	0.0	0.
Soil_Type14	80000.0	0.014788	0.120702	0.0	0.0	0.
Soil_Type15	80000.0	0.000000	0.000000	0.0	0.0	0.
Soil_Type16	80000.0	0.015988	0.125428	0.0	0.0	0.
Soil_Type17	80000.0	0.020737	0.142505	0.0	0.0	0.
Soil_Type18	80000.0	0.013300	0.114557	0.0	0.0	0.
Soil_Type19	80000.0	0.014087	0.117853	0.0	0.0	0.
Soil_Type20	80000.0	0.017487	0.131080	0.0	0.0	0.
Soil_Type21	80000.0	0.012125	0.109445	0.0	0.0	0.
Soil_Type22	80000.0	0.031288	0.174095	0.0	0.0	0.
Soil_Type23	80000.0	0.049875	0.217688	0.0	0.0	0.
Soil_Type24	80000.0	0.024375	0.154211	0.0	0.0	0.
Soil_Type25	80000.0	0.003150	0.056037	0.0	0.0	0.
Soil_Type26	80000.0	0.013225	0.114238	0.0	0.0	0.
Soil_Type27	80000.0	0.011612	0.107134	0.0	0.0	0.
Soil_Type28	80000.0	0.010988	0.104244	0.0	0.0	0.
Soil_Type29	80000.0	0.021862	0.146235	0.0	0.0	0.
Soil_Type30	80000.0	0.028587	0.166645	0.0	0.0	0.
Soil_Type31	80000.0	0.027125	0.162449	0.0	0.0	0.
Soil_Type32	80000.0	0.038150	0.191559	0.0	0.0	0.
Soil_Type33	80000.0	0.037687	0.190441	0.0	0.0	0.
Soil_Type34	80000.0	0.011838	0.108155	0.0	0.0	0.
Soil_Type35	80000.0	0.015425	0.123237	0.0	0.0	0.
Soil_Type36	80000.0	0.010812	0.103420	0.0	0.0	0.
Soil_Type37	80000.0	0.012538	0.111268	0.0	0.0	0.
Soil_Type38	80000.0	0.040325	0.196722	0.0	0.0	0.
Soil_Type39	80000.0	0.039163	0.193983	0.0	0.0	0.
Soil_Type40	80000.0	0.030437	0.171789	0.0	0.0	0.

In [154]:

```
Y= df.Cover_Type.values
df.drop(columns = ['Cover_Type'], inplace = True)
```

Q2: Observe the range of all feature values from the dataframe description above.

1. Do you think in our dataset normalization is required? -- Give proper justification based on

your opinion.
2. What type of normalization/Scaling technique you whould recommend for our dataset?

A2

`Answer 1:` In my opinion, I would normalize the dataset due to some of the very numbers we seeing in standard deviation, mean, and min/max. We can see that the values in the variables 'horizontal_distance' has this set of attributes which means we have an abnormal distribution of data and normalizing the data should yield a better output.

`Answer 2:` To fix this issue I will be using standard scalar from the sklearn librabry. Standard scalar is a scaling method.

Q3:

1. Use the above mentioned normalization technique on our HW_8 dataset.
2. Transform the X dataframe using choosen normalization technique.

`Note:` Make sure the scaled X has all column name same as `X dataframe`

A3 Fill the cell blocks below, Create new cell as per your necessary

In [155]:
```
sc = StandardScaler()
sc.fit(X)
Scaled_X = sc.fit_transform(X)
```

Out[155]:
```
(80000, 54)
```

In [156]:
```
xtrain, xtest, ytrain, ytest =  train_test_split(Scaled_X, Y, test_size = 
```

Q4:

1. Check and show if there is any null values in our dataset.
2. Print all unique values/ different class id from the `Y data` .

A4 Fill the cell blocks below, Create new cell as per your necessary

In [157]:

Out[157]:

```
Elevation                             0
Aspect                                0
Slope                                 0
Horizontal_Distance_To_Hydrology      0
Vertical_Distance_To_Hydrology        0
Horizontal_Distance_To_Roadways       0
Hillshade_9am                         0
Hillshade_Noon                        0
Hillshade_3pm                         0
Horizontal_Distance_To_Fire_Points    0
Wilderness_Area1                      0
Wilderness_Area2                      0
Wilderness_Area3                      0
Wilderness_Area4                      0
Soil_Type1                            0
Soil_Type2                            0
Soil_Type3                            0
Soil_Type4                            0
Soil_Type5                            0
Soil_Type6                            0
Soil_Type7                            0
Soil_Type8                            0
Soil_Type9                            0
Soil_Type10                           0
Soil_Type11                           0
Soil_Type12                           0
Soil_Type13                           0
Soil_Type14                           0
Soil_Type15                           0
Soil_Type16                           0
Soil_Type17                           0
Soil_Type18                           0
Soil_Type19                           0
Soil_Type20                           0
Soil_Type21                           0
Soil_Type22                           0
Soil_Type23                           0
Soil_Type24                           0
Soil_Type25                           0
Soil_Type26                           0
Soil_Type27                           0
Soil_Type28                           0
Soil_Type29                           0
```

In [158]:

Out[158]:
```
array([1, 2, 3, 7, 6, 4], dtype=int64)
```

Part 1: Use a subset of whole data(N=20000) for Data Visualization

Data Subset Creation

1. First we are Selecting `N=20000` random rows from our dataset and create a new subset of data.

In [176]:
```
#converted Scaled_X to a dataframe to be able to run the ".loc" function
Scaled_Xdf = pd.DataFrame(Scaled_X, columns = df.columns)
Ydf = pd.DataFrame(Y, columns = ['Cover_Type'])
```
Out[176]:
```
(80000, 54)
```

In [177]:
```
np.random.seed(42)
rndperm = np.random.permutation(df.shape[0])
N = 20000
data_subset_x = Scaled_Xdf.loc[rndperm[:N],:].copy()
data_subset_y = Ydf.loc[rndperm[:N]].copy()

data_subset = data_subset_x.values
```
Out[177]:
```
(20000, 54)
```

Q5:

1. Use PCA and reduce the dimension of the **data_subset_x** into `3`.
2. Store the PCA reuslt into `pca_result` variable
3. Add the `3` PCA reduced columns into **data_subset_x**

A5 Fill the below cells. Use extra cells as per your necessary

In [170]:
```
pca = PCA(n_components = 3)
pca_result = pca.fit_transform(data_subset)
```
Out[170]:
```
(20000, 3)
```

In [171]:
```
data_subset_x['pca-one'] = pca_result[:,0]
data_subset_x['pca-two'] = pca_result[:,1]
```

Q6:

1. Use TSNE and reduce the dimension of the **data_subset_x** into `2`.
2. Store the TSNE reuslt into `tsne_results` variable
3. Add the `2` TSNE reduced columns into **data_subset_x**

`Note:`

1. You can use `from sklearn.manifold import TSNE` for TSNE initialization.
2. Give value of n_components as per the question.
3. Also use other parameters while TSNE initialization as, `verbose=1, perplexity=40, n_iter=300`

A6 Fill the below cells. Use extra cells as per your necessary

In [172]:
```
tsne = TSNE(n_components = 2,verbose = 1, perplexity = 40, n_iter = 300)
```

```
C:\Users\pedro\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:78
0: FutureWarning: The default initialization in TSNE will change from 'ra
ndom' to 'pca' in 1.2.
  warnings.warn(
C:\Users\pedro\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:79
0: FutureWarning: The default learning rate in TSNE will change from 200.
0 to 'auto' in 1.2.
  warnings.warn(

[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 20000 samples in 0.000s...
[t-SNE] Computed neighbors for 20000 samples in 10.340s...
[t-SNE] Computed conditional probabilities for sample 1000 / 20000
[t-SNE] Computed conditional probabilities for sample 2000 / 20000
[t-SNE] Computed conditional probabilities for sample 3000 / 20000
[t-SNE] Computed conditional probabilities for sample 4000 / 20000
[t-SNE] Computed conditional probabilities for sample 5000 / 20000
[t-SNE] Computed conditional probabilities for sample 6000 / 20000
[t-SNE] Computed conditional probabilities for sample 7000 / 20000
[t-SNE] Computed conditional probabilities for sample 8000 / 20000
[t-SNE] Computed conditional probabilities for sample 9000 / 20000
[t-SNE] Computed conditional probabilities for sample 10000 / 20000
[t-SNE] Computed conditional probabilities for sample 11000 / 20000
[t-SNE] Computed conditional probabilities for sample 12000 / 20000
[t-SNE] Computed conditional probabilities for sample 13000 / 20000
[t-SNE] Computed conditional probabilities for sample 14000 / 20000
[t-SNE] Computed conditional probabilities for sample 15000 / 20000
[t-SNE] Computed conditional probabilities for sample 16000 / 20000
[t-SNE] Computed conditional probabilities for sample 17000 / 20000
[t-SNE] Computed conditional probabilities for sample 18000 / 20000
[t-SNE] Computed conditional probabilities for sample 19000 / 20000
[t-SNE] Computed conditional probabilities for sample 20000 / 20000
[t-SNE] Mean sigma: 1.247504
[t-SNE] KL divergence after 250 iterations with early exaggeration: 77.51
0605
[t-SNE] KL divergence after 300 iterations: 3.183555
```

In [173]:
```
data_subset_x['tsne-2d-one'] = tsne_results[:,0]
```

In [ ]:
```
```

Q7:

1. Create a new dataframe with name `df_plot`
2. This dataframe whill merge everything from **data_subset_x** and **data_subset_y**
3. We need to give a name for the `data_subset_y` column. Use `Cover_Type` as the name of the column

A7 Fill the below cells. Use extra cells as per your necessary

In [178]:
```
df_plot= pd.concat([data_subset_x,data_subset_y],ignore_index=True, sort=F
```

In [179]:
```
df_plot.head()
```

Out[179]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_H
0	-1.678761	-0.933505	0.809499	-0.121145	-
1	-0.762002	-0.042150	-0.245439	-0.912247	-
2	-0.748111	0.749155	-0.011008	-0.903457	-
3	-0.400854	-1.233655	-0.362654	-0.547461	-
4	-1.595419	-0.033054	0.106207	-0.402426	

5 rows × 55 columns

Q8: Now we will plot all points from our dataframe `df_plot` Using the result from **PCA**

1. Use `pca-one` and `pca-two` column as X and Y axis respectively.
2. Use seaborn scatterplot for plotting the points.

`Note:` Use the notebook from class 4/11 for data plotting. The link is provided below.

`Link:` [https://git.txstate.edu/ML/2022Spring/blob/master/project/Data_Viz_with_PCA_TSNE.ipynb (https://git.txstate.edu/ML/2022Spring/blob/master/project/Data_Viz_with_PCA_TSNE.ipynb)](https://git.txstate.edu/ML/2022Spring/blob/master/project/Data_Viz_with_PCA_TSNE.ipynb)

A8 Fill the below cells. Use extra cells as per your necessary

In [ ]:
```
plt.figure(figsize=(16,10))
sns.scatterplot(

)
```

Q9: Now we will plot all points from our dataframe `df_plot` Using result from T-SNE.

1. Use `tsne-2d-one` and `tsne-2d-one` column as X and Y axis respectively.
2. Use seaborn scatterplot for plotting the points.

`Note:` Use the notebook from class 4/11 for data plotting. The link is provided below.

`Link:` [https://git.txstate.edu/ML/2022Spring/blob/master/project/Data_Viz_with_PCA_TSNE.ipynb (https://git.txstate.edu/ML/2022Spring/blob/master/project/Data_Viz_with_PCA_TSNE.ipynb)](https://git.txstate.edu/ML/2022Spring/blob/master/project/Data_Viz_with_PCA_TSNE.ipynb)

A9 Fill the below cells. Use extra cells as per your necessary

In [ ]:
```
plt.figure(figsize=(16,10))
sns.scatterplot(

)
```

Part 2: Data Analysis and Classification Using Entire Dataset

Q10: Observe the data plotting and find the realtion between datapoints and their characteristics.

1. Reduce the dimension of our `Scaled_X` dataframe to `3` using PCA algorithm.
2. Store the result into a variable named `pca_result`
3. Create Train data and Test data using the pca_result and Y.

```
Note:
```

1. Consider pca_result as X values, and Y as y values.
2. You can use sklearn train_test_split
3. Keep Train and Test ratio as : 75%:25%

A10 Fill the below cells. Use extra cells as per your necessary

In [ ]:
```
pca =
```

In [ ]:
```
```

Now, Select Three best model for our dataset. You have to decide three models which might work well with our dataset.

Q11

Model Number 1

1. Reason behind choosing the model.
2. Create the model using sklearn or any proper library
3. Fit the model with the train data
4. Get the score from the model using test data

A11 Fill the below cells. Use extra cells as per your necessary

```
Answer for Q.No:1 goes here
```

In [ ]:
```
```

Q12

Model Number 2

1. Reason behind choosing the model.
2. Create the model using sklearn or any proper library
3. Fit the model with the train data

4. Get the score from the model using test data

A12 Fill the below cells. Use extra cells as per your necessaryReplace ??? with code in the code cell below

```
Answer for Q.No:1 goes here
```

In [ ]:

Q13

Model Number 3

1. Reason behind choosing the model.
2. Create the model using sklearn or any proper library
3. Fit the model with the train data
4. Get the score from the model using test data

A13 Fill the below cells. Use extra cells as per your necessary

```
Answer for Q.No:1 goes here
```

In [ ]:

Q14

1. Plot a histogram using Y dataframe and display the per-class data distribution(number of rows per class).
2. Also print the number of rows per class as numeric value.

A14 Fill the below cells. Use extra cells as per your necessary

In [ ]:

In [ ]:

Q15

1. From the histogram we can see that the dataset is highly imbalanced.
2. Use a proper dataset balancing technique to make the dataset balanced.
3. Plot a histogram using new y values and display the per-class data distribution(number of rows per class).

`Note:` Use can use the `imblearn.over_sampling` library for this task. But use appropriate strategy for the method.

Follow the documentation for details: [https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html (https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html)](https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html)

A15 Fill the below cells. Use extra cells as per your necessary

In [ ]:

In [ ]:

Q16

1. Create new Train and Test data from the balaned X and Y value.
2. Keep Train and Test ratio as : 75%:25%

A16 Fill the below cells. Use extra cells as per your necessary

In [ ]:

Q17

Now, Use the previously initialized three models and calculate the score from our new balanced dataset.

Model Number 1

1. Fit the model with the new train data(Use the previous Model 1)
2. Get the score from the model using new test data

A17 Fill the below cells. Use extra cells as per your necessary

In [ ]:

Model Number 2

1. Fit the model with the new train data(Use the previous Model 2)
2. Get the score from the model using new test data

Fill the below cells. Use extra cells as per your necessary

In [ ]:

Model Number 3

1. Fit the model with the new train data(Use the previous Model 3)
2. Get the score from the model using new test data

Fill the below cells. Use extra cells as per your necessary

In [ ]:

After making the dataset balanced we can see a significant improve in the performence for all three models.