In [ ]:
```
##### Author : Amir Shokri
##### github link : https://github.com/amirshnll/Covertype
##### dataset link : http://archive.ics.uci.edu/ml/datasets/Covertype
##### email : amirsh.nll@gmail.com
```

In [1]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import classification_report
```

In [2]:
```
df = pd.read_csv('covtype_data.csv', header=None)
```

In [3]:
```
df
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	45	46	47	48	49	50	51
0	2596	51	3	258	0	510	221	232	148	6279	...	0	0	0	0	0	0	0
1	2590	56	2	212	-6	390	220	235	151	6225	...	0	0	0	0	0	0	0
2	2804	139	9	268	65	3180	234	238	135	6121	...	0	0	0	0	0	0	0
3	2785	155	18	242	118	3090	238	238	122	6211	...	0	0	0	0	0	0	0
4	2595	45	2	153	-1	391	220	234	150	6172	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
581007	2396	153	20	85	17	108	240	237	118	837	...	0	0	0	0	0	0	0
581008	2391	152	19	67	12	95	240	237	119	845	...	0	0	0	0	0	0	0
581009	2386	159	17	60	7	90	236	241	130	854	...	0	0	0	0	0	0	0
581010	2384	170	15	60	5	90	230	245	143	864	...	0	0	0	0	0	0	0
581011	2383	165	13	60	4	67	231	244	141	875	...	0	0	0	0	0	0	0

581012 rows × 55 columns

In [4]:
```
df.describe()
```

Out[4]:

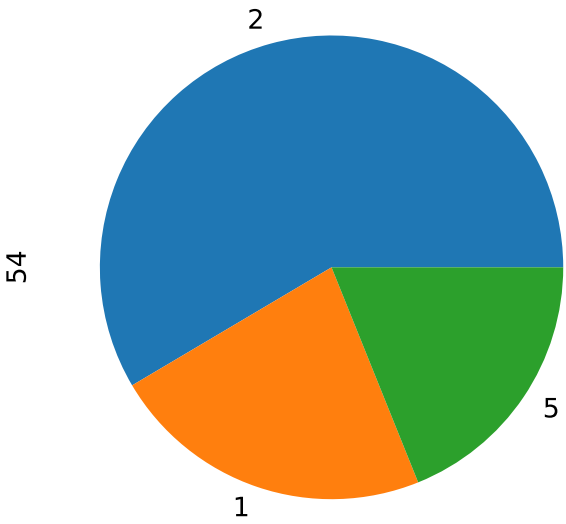
	0	1	2	3	4	
count	581012.000000	581012.000000	581012.000000	581012.000000	581012.000000	581012.00000
mean	2959.365301	155.656807	14.103704	269.428217	46.418855	2350.1466
std	279.984734	111.913721	7.488242	212.549356	58.295232	1559.25487
min	1859.000000	0.000000	0.000000	0.000000	-173.000000	0.00000
25%	2809.000000	58.000000	9.000000	108.000000	7.000000	1106.00000
50%	2996.000000	127.000000	13.000000	218.000000	30.000000	1997.00000
75%	3163.000000	260.000000	18.000000	384.000000	69.000000	3328.00000
max	3858.000000	360.000000	66.000000	1397.000000	601.000000	7117.00000

8 rows × 55 columns

In [13]:
```
x = df[df.columns[:54]]
y = df[df.columns[54]]
scaler = MinMaxScaler()
scaled_x = scaler.fit_transform(x)
```

In [14]:
```
y.value_counts().plot.pie()
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c9c8d1c488>
```



In [6]:
```
#Dimentionality reduction
pca = PCA(n_components=15)
reduced_x = pca.fit_transform(scaled_x)
```

In [7]:
```
#Choose whether reduces or not
X = scaled_x
X = reduced_x
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
m_state=0)
```

In [10]:
```
#Now we run algorithms and evaluate
```

In [11]:
```
from sklearn.naive_bayes import CategoricalNB
cnb = CategoricalNB()
cnb.fit(X_train, y_train)
predicted = cnb.predict(X_test)

print('MSE:', MSE(y_test, predicted))
print(classification_report(y_test, predicted))
```

```
MSE: 1.8519196346612814
              precision    recall  f1-score   support

           1       0.67      0.03      0.06     63498
           2       0.51      0.98      0.67     85198
           3       0.58      0.33      0.42     10581
           4       0.00      0.00      0.00       822
           5       0.00      0.00      0.00      2850
           6       0.00      0.00      0.00      5229
           7       0.00      0.00      0.00      6126

    accuracy                           0.51    174304
   macro avg       0.25      0.19      0.16    174304
weighted avg       0.52      0.51      0.37    174304
```

In [10]:
```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(100, 100), activation='relu', solver=
'adam', alpha=0.0001)
mlp.fit(X_train, y_train)
predicted = mlp.predict(X_test)

print('MSE:', MSE(y_test, predicted))
print(classification_report(y_test, predicted))
```

```
MSE: 0.96
              precision    recall  f1-score   support

           1       0.66      0.58      0.62        71
           2       0.81      0.82      0.81       168
           5       0.78      0.87      0.82        61

    accuracy                           0.77       300
   macro avg       0.75      0.75      0.75       300
weighted avg       0.77      0.77      0.77       300
```

In [15]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
predicted = knn.predict(X_test)

print('MSE:', MSE(y_test, predicted))
print(classification_report(y_test, predicted))
```

```
MSE: 0.35506930420414906
              precision    recall  f1-score   support

           1       0.93      0.92      0.93     63498
           2       0.94      0.95      0.94     85198
           3       0.92      0.93      0.92     10581
           4       0.86      0.71      0.78       822
           5       0.84      0.77      0.80      2850
           6       0.87      0.86      0.87      5229
           7       0.94      0.93      0.93      6126

    accuracy                           0.93    174304
   macro avg       0.90      0.87      0.88    174304
weighted avg       0.93      0.93      0.93    174304
```

In [13]:
```
from sklearn.tree import DecisionTreeClassifier
Dtree = DecisionTreeClassifier()
Dtree.fit(X_train, y_train)
predicted = Dtree.predict(X_test)

print('MSE:', MSE(y_test, predicted))
print(classification_report(y_test, predicted))
```

```
MSE: 0.4798799798053975
              precision    recall  f1-score   support

           1       0.90      0.90      0.90     63498
           2       0.91      0.91      0.91     85198
           3       0.88      0.88      0.88     10581
           4       0.74      0.73      0.74       822
           5       0.73      0.74      0.74      2850
           6       0.80      0.80      0.80      5229
           7       0.92      0.92      0.92      6126

    accuracy                           0.90    174304
   macro avg       0.84      0.84      0.84    174304
weighted avg       0.90      0.90      0.90    174304
```

In [14]:
```
from sklearn.linear_model import LogisticRegression
lreg = LogisticRegression()
lreg.fit(X_train, y_train)
predicted = lreg.predict(X_test)

print('MSE:', MSE(y_test, predicted))
print(classification_report(y_test, predicted))
```

```
MSE: 1.7821335138608407
              precision    recall  f1-score   support

           1       0.66      0.57      0.61     63498
           2       0.67      0.80      0.73     85198
           3       0.60      0.76      0.67     10581
           4       0.11      0.00      0.00       822
           5       1.00      0.00      0.00      2850
           6       0.37      0.10      0.15      5229
           7       0.66      0.37      0.47      6126

    accuracy                           0.66    174304
   macro avg       0.58      0.37      0.38    174304
weighted avg       0.66      0.66      0.64    174304
```

In [ ]: