Project - Forest Cover

Import libraries

In [1]:
```
import numpy as np
import pandas as pd

from IPython.display import display
pd.options.display.max_columns = None
```

Import Data

In [2]:
```
data = pd.read_csv('train.csv')
```

Process Data

In [3]:
```
data.head(5)
```

Out[3]:

	Id	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrolo
0	1	2596	51	3	258	
1	2	2590	56	2	212	
2	3	2804	139	9	268	
3	4	2785	155	18	242	1
4	5	2595	45	2	153	

In [4]:
```
types = data['Cover_Type']
types.head(5)
```

Out[4]:
```
0    5
1    5
2    2
3    2
4    5
Name: Cover_Type, dtype: int64
```

In [5]:
```
# types = np.array(types)
```

In [6]:
```
features = data.drop('Cover_Type', axis=1)
features = features.drop('Id', axis=1)
features.head(5)
```

Out[6]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology
0	2596	51	3	258	0
1	2590	56	2	212	-6
2	2804	139	9	268	65
3	2785	155	18	242	118
4	2595	45	2	153	-1

In [7]:
```
# features = np.array(features)
```

In [8]:
```
names = list(features)
```

Explore Data

In [9]:
```
data.describe()
```

Out[9]:

	Id	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrolog
count	15120.00000	15120.000000	15120.000000	15120.000000	15120.00000
mean	7560.50000	2749.322553	156.676653	16.501587	227.19570
std	4364.91237	417.678187	110.085801	8.453927	210.07529
min	1.00000	1863.000000	0.000000	0.000000	0.00000
25%	3780.75000	2376.000000	65.000000	10.000000	67.00000
50%	7560.50000	2752.000000	126.000000	15.000000	180.00000
75%	11340.25000	3104.000000	261.000000	22.000000	330.00000
max	15120.00000	3849.000000	360.000000	52.000000	1343.00000

In [10]:
```
list(data)
```

Out[10]:
```
['Id',
 'Elevation',
 'Aspect',
 'Slope',
 'Horizontal_Distance_To_Hydrology',
 'Vertical_Distance_To_Hydrology',
 'Horizontal_Distance_To_Roadways',
 'Hillshade_9am',
 'Hillshade_Noon',
 'Hillshade_3pm',
 'Horizontal_Distance_To_Fire_Points',
 'Wilderness_Area1',
 'Wilderness_Area2',
 'Wilderness_Area3',
 'Wilderness_Area4',
 'Soil_Type1',
 'Soil_Type2',
 'Soil_Type3',
 'Soil_Type4',
 'Soil_Type5',
 'Soil_Type6',
 'Soil_Type7',
 'Soil_Type8',
 'Soil_Type9',
 'Soil_Type10',
 'Soil_Type11',
 'Soil_Type12',
 'Soil_Type13',
 'Soil_Type14',
 'Soil_Type15',
 'Soil_Type16',
 'Soil_Type17',
 'Soil_Type18',
 'Soil_Type19',
 'Soil_Type20',
 'Soil_Type21',
 'Soil_Type22',
 'Soil_Type23',
 'Soil_Type24',
 'Soil_Type25',
 'Soil_Type26',
 'Soil_Type27',
 'Soil_Type28',
 'Soil_Type29',
 'Soil_Type30',
 'Soil_Type31',
 'Soil_Type32',
 'Soil_Type33',
 'Soil_Type34',
 'Soil_Type35',
 'Soil_Type36',
 'Soil_Type37',
 'Soil_Type38',
 'Soil_Type39',
 'Soil_Type40',
 'Cover_Type']
```

In [59]:
```
%matplotlib inline
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = (30,30)
```

In [60]:
```
plt.subplot(5,2,1)
plt.scatter(data['Elevation'], data['Cover_Type'])
plt.subplot(5,2,2)
plt.scatter(data['Aspect'], data['Cover_Type'])
plt.subplot(5,2,3)
plt.scatter(data['Slope'], data['Cover_Type'])
plt.subplot(5,2,4)
plt.scatter(data['Horizontal_Distance_To_Hydrology'], data['Cover_Typ
e'])
plt.subplot(5,2,5)
plt.scatter(data['Vertical_Distance_To_Hydrology'], data['Cover_Type'
])
plt.subplot(5,2,6)
plt.scatter(data['Horizontal_Distance_To_Roadways'], data['Cover_Type'
])
plt.subplot(5,2,7)
plt.scatter(data['Hillshade_9am'], data['Cover_Type'])
plt.subplot(5,2,8)
plt.scatter(data['Hillshade_Noon'], data['Cover_Type'])
plt.subplot(5,2,9)
plt.scatter(data['Hillshade_3pm'], data['Cover_Type'])
plt.subplot(5,2,10)
plt.scatter(data['Horizontal_Distance_To_Fire_Points'], data['Cover_Ty
pe'])
```

Out[60]: <matplotlib.collections.PathCollection at 0x7f82b9bb4c18>

Final Data Type Stuff

In [13]:
```
typeCats = types.astype('category')
```

Test/Train Split

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
train_features, test_features, train_types, test_types = train_test_sp
lit(features, types, test_size = 0.15, random_state = 42)
```

In [16]:
```
train_features, test_features, train_typeCats, test_typeCats = train_t
est_split(features, typeCats, test_size = 0.15, random_state = 42)
```

Linear Regression

In [17]:
```
from sklearn.linear_model import LinearRegression
```

In [18]:
```
linReg = LinearRegression().fit(train_features, train_typeCats)
```

In [19]:
```
coeffs = linReg.coef_
indices1 = np.argsort(coeffs)[::-1]
```

In [20]:
```
for f in range(features.shape[1]):
    print("%d. feature %d (%f) %s" % (f + 1, indices1[f], coeffs[indices1[f]], names[indices1[f]]))
```

```
1. feature 50 (4.202742) Soil_Type37
2. feature 48 (3.236726) Soil_Type35
3. feature 53 (3.158550) Soil_Type40
4. feature 51 (2.938677) Soil_Type38
5. feature 49 (2.650143) Soil_Type36
6. feature 52 (2.632918) Soil_Type39
7. feature 43 (0.721170) Soil_Type30
8. feature 12 (0.653278) Wilderness_Area3
9. feature 31 (0.497356) Soil_Type18
10. feature 23 (0.347611) Soil_Type10
11. feature 13 (0.199109) Wilderness_Area4
12. feature 18 (0.138926) Soil_Type5
13. feature 26 (0.123921) Soil_Type13
14. feature 29 (0.045109) Soil_Type16
15. feature 47 (0.044812) Soil_Type34
16. feature 6 (0.017650) Hillshade_9am
17. feature 8 (0.010914) Hillshade_3pm
18. feature 2 (0.007294) Slope
19. feature 4 (0.001703) Vertical_Distance_To_Hydrology
20. feature 1 (0.000641) Aspect
21. feature 9 (0.000101) Horizontal_Distance_To_Fire_Points
22. feature 20 (0.000000) Soil_Type7
23. feature 28 (-0.000000) Soil_Type15
24. feature 5 (-0.000155) Horizontal_Distance_To_Roadways
25. feature 0 (-0.000504) Elevation
26. feature 3 (-0.001003) Horizontal_Distance_To_Hydrology
27. feature 7 (-0.013403) Hillshade_Noon
28. feature 30 (-0.031646) Soil_Type17
29. feature 27 (-0.068114) Soil_Type14
30. feature 15 (-0.164524) Soil_Type2
31. feature 19 (-0.168584) Soil_Type6
32. feature 14 (-0.178935) Soil_Type1
33. feature 24 (-0.230781) Soil_Type11
34. feature 11 (-0.238037) Wilderness_Area2
35. feature 42 (-0.318591) Soil_Type29
36. feature 32 (-0.495734) Soil_Type19
37. feature 16 (-0.580552) Soil_Type3
38. feature 10 (-0.614350) Wilderness_Area1
39. feature 17 (-0.702509) Soil_Type4
40. feature 21 (-0.727258) Soil_Type8
41. feature 33 (-0.812114) Soil_Type20
42. feature 46 (-0.850617) Soil_Type33
43. feature 36 (-0.887840) Soil_Type23
44. feature 39 (-0.899927) Soil_Type26
45. feature 40 (-1.094524) Soil_Type27
46. feature 41 (-1.133955) Soil_Type28
47. feature 44 (-1.177063) Soil_Type31
48. feature 25 (-1.183532) Soil_Type12
49. feature 45 (-1.211052) Soil_Type32
50. feature 37 (-1.351876) Soil_Type24
51. feature 22 (-1.383956) Soil_Type9
52. feature 34 (-1.635531) Soil_Type21
53. feature 38 (-1.713170) Soil_Type25
54. feature 35 (-1.736275) Soil_Type22
```

In [21]:
```
linReg.score(test_features, test_typeCats)
```
Out[21]: 0.4214392425805952

In [22]:
```
linReg.score(train_features, train_typeCats)
```
Out[22]: 0.4006483305915349

In [23]:
```
from sklearn.metrics import mean_squared_error
```

In [24]:
```
mean_squared_error(train_typeCats, linReg.predict(train_features))
```
Out[24]: 2.3952531121506153

In [25]:
```
mean_squared_error(test_typeCats, linReg.predict(test_features))
```
Out[25]: 2.3257183606210883

Logistic Regression

In [26]:
```
from sklearn.linear_model import LogisticRegression
```

In [27]:
```
logReg1 = LogisticRegression(random_state=0, \
                             solver='lbfgs', \
                             multi_class='ovr', \
                             max_iter = 10000, \
                             n_jobs = 6).fit(train_features, train_type
Cats)
```

In [28]:
```
logReg2 = LogisticRegression(random_state=0, \
                             solver='lbfgs', \
                             multi_class='multinomial', \
                             max_iter = 10000, \
                             n_jobs = 6).fit(train_features, train_type
Cats)
```

In [29]:
```
logReg1.score(test_features, test_typeCats)
```
Out[29]: 0.6710758377425045

In [30]:
```
logReg1.score(train_features, train_typeCats)
```
Out[30]: 0.6718798630563336

In [31]:
```
logReg2.score(test_features, test_typeCats)
```
Out[31]: 0.6653439153439153

In [32]:
```
logReg2.score(train_features, train_typeCats)
```
Out[32]: 0.6721132897603486

In [33]:
```
mean_squared_error(train_typeCats, logReg1.predict(train_features))
```
Out[33]: 2.9991441020852787

In [34]:
```
mean_squared_error(test_typeCats, logReg1.predict(test_features))
```
Out[34]: 2.873456790123457

In [35]:
```
mean_squared_error(train_typeCats, logReg2.predict(train_features))
```
Out[35]: 3.0791316526610646

In [36]:
```
mean_squared_error(test_typeCats, logReg2.predict(test_features))
```
Out[36]: 3.2865961199294533

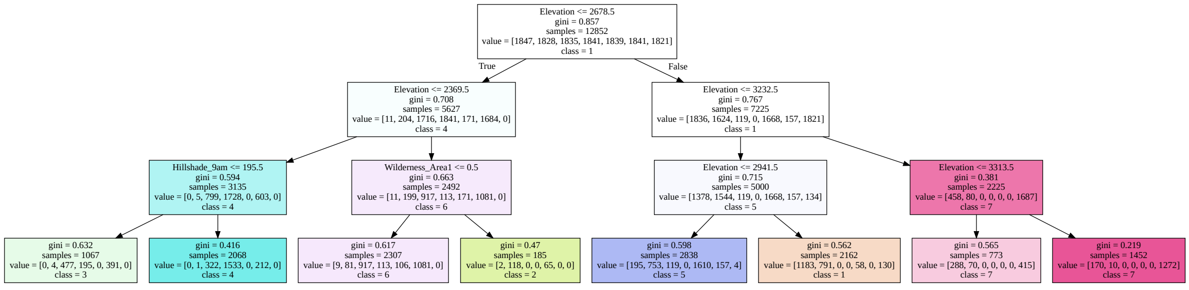
Decision Tree

In [37]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [38]:
```
decTreePartial1 = DecisionTreeClassifier(random_state=0, max_depth=3).
fit(train_features, train_typeCats)
```

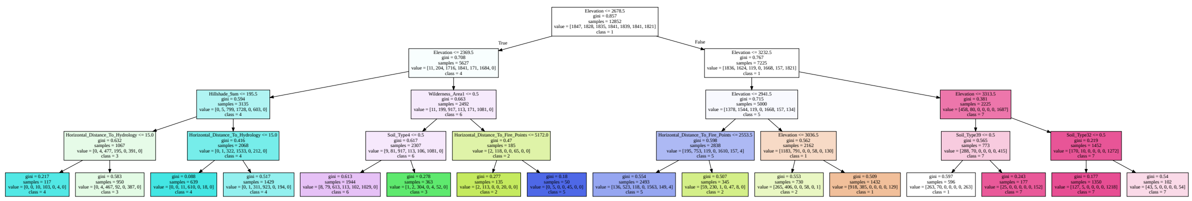
In [39]:
```
from sklearn import tree
from IPython.display import SVG
from graphviz import Source
from IPython.display import display
```

In [40]:
```
graph1 = Source(tree.export_graphviz(decTreePartial1, out_file=None, f
eature_names=names, class_names=['1','2','3','4','5','6','7'], filled=
True))
graph1a = Source(tree.export_graphviz(decTreePartial1, out_file='decTr
ee1.dot', feature_names=names, class_names=['1','2','3','4','5','6',
'7'], filled=True))

display(SVG(graph1.pipe(format='svg')))
```



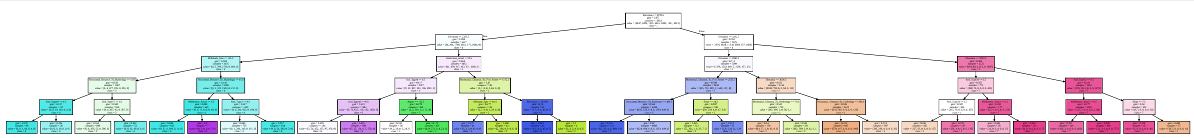
In [41]:
```
decTreePartial2 = DecisionTreeClassifier(random_state=0, max_depth=4).fit(train_features, train_typeCats)
```

In [42]:
```
graph2 = Source(tree.export_graphviz(decTreePartial2, out_file=None, feature_names=names, class_names=['1','2','3','4','5','6','7'], filled=True))
graph2a = Source(tree.export_graphviz(decTreePartial2, out_file='decTree2.dot', feature_names=names, class_names=['1','2','3','4','5','6','7'], filled=True))

display(SVG(graph2.pipe(format='svg')))
```



In [63]:
```
decTreePartial3 = DecisionTreeClassifier(random_state=0, max_depth=5).fit(train_features, train_typeCats)
```

In [66]:
```
graph3 = Source(tree.export_graphviz(decTreePartial3, out_file=None, feature_names=names, class_names=['1','2','3','4','5','6','7'], filled=True))
graph3a = Source(tree.export_graphviz(decTreePartial3, out_file='decTree3.dot', feature_names=names, class_names=['1','2','3','4','5','6','7'], filled=True))

display(SVG(graph3.pipe(format='svg')))
```



In [43]:
```
decTreeFull = DecisionTreeClassifier(random_state=0).fit(train_features, train_typeCats)
graph = Source(tree.export_graphviz(decTreeFull, out_file='decTree.dot', feature_names=names, class_names=['1','2','3','4','5','6','7'], filled=True))
```

In [44]:
```
decTreeFull.score(test_features, test_typeCats)
```

Out[44]:
```
0.8077601410934744
```

In [45]:
```
mean_squared_error(test_typeCats, decTreeFull.predict(test_features))
```

Out[45]:
```
1.7570546737213404
```

Random Forest

In [46]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [47]:
```
randFor = RandomForestClassifier(n_estimators=10000, n_jobs=6, random_
state=43).fit(train_features, train_typeCats)
```

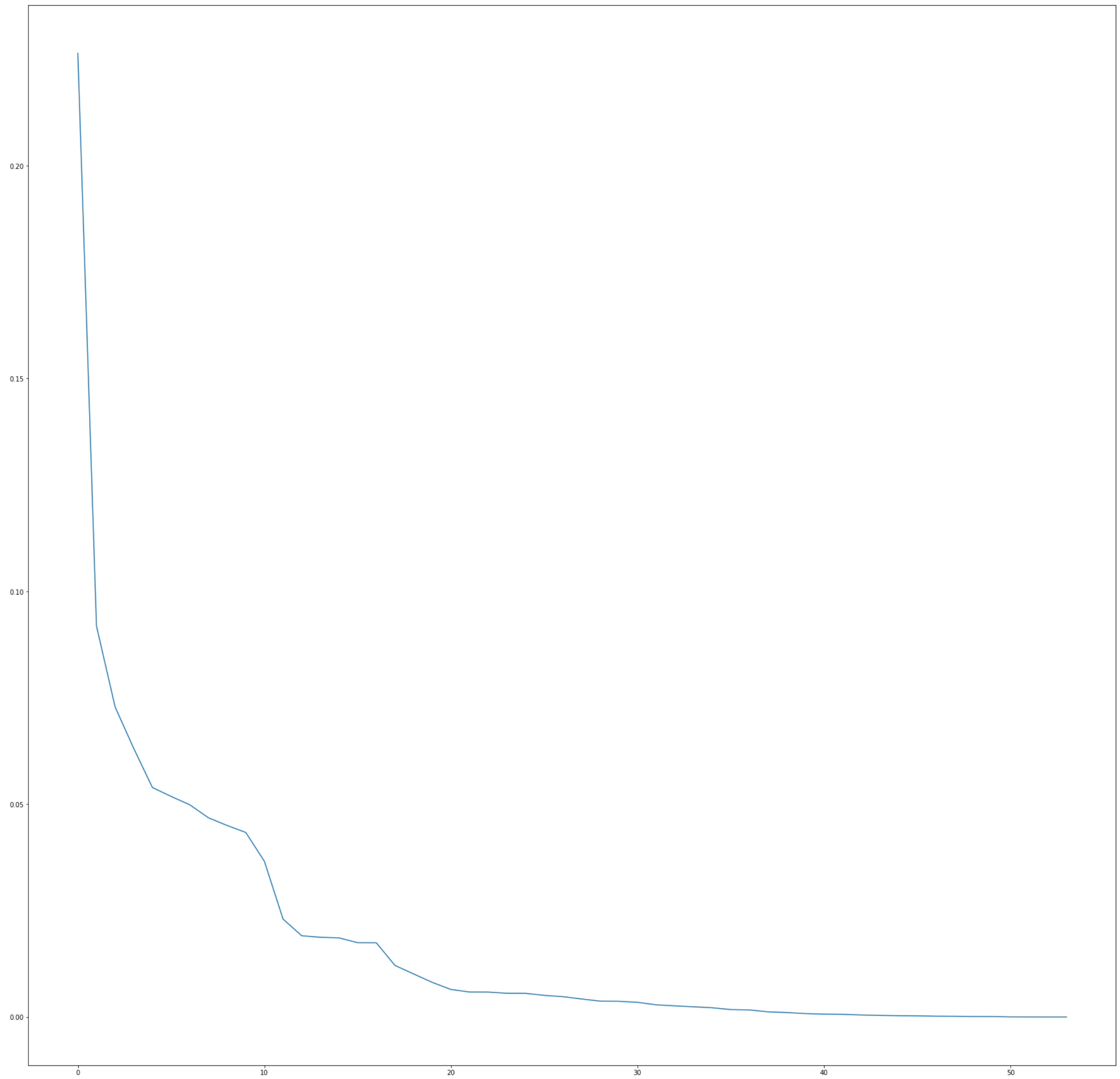
In [48]:
```
importances = randFor.feature_importances_
indices2 = np.argsort(importances)[::-1]
```

In [49]:
```
for f in range(features.shape[1]):
    print("%d. feature %d (%f) %s" % (f + 1, indices2[f], importances[
indices2[f]], names[indices2[f]]))
```

```
1. feature 0 (0.226407) Elevation
2. feature 5 (0.091830) Horizontal_Distance_To_Roadways
3. feature 9 (0.072816) Horizontal_Distance_To_Fire_Points
4. feature 3 (0.063058) Horizontal_Distance_To_Hydrology
5. feature 4 (0.053903) Vertical_Distance_To_Hydrology
6. feature 6 (0.051818) Hillshade_9am
7. feature 1 (0.049855) Aspect
8. feature 8 (0.046800) Hillshade_3pm
9. feature 7 (0.044994) Hillshade_Noon
10. feature 13 (0.043361) Wilderness_Area4
11. feature 2 (0.036584) Slope
12. feature 23 (0.023007) Soil_Type10
13. feature 51 (0.019082) Soil_Type38
14. feature 10 (0.018722) Wilderness_Area1
15. feature 16 (0.018589) Soil_Type3
16. feature 52 (0.017449) Soil_Type39
17. feature 12 (0.017432) Wilderness_Area3
18. feature 17 (0.012110) Soil_Type4
19. feature 53 (0.010111) Soil_Type40
20. feature 43 (0.008126) Soil_Type30
21. feature 15 (0.006463) Soil_Type2
22. feature 30 (0.005861) Soil_Type17
23. feature 26 (0.005849) Soil_Type13
24. feature 35 (0.005563) Soil_Type22
25. feature 42 (0.005552) Soil_Type29
26. feature 36 (0.005078) Soil_Type23
27. feature 45 (0.004771) Soil_Type32
28. feature 25 (0.004231) Soil_Type12
29. feature 11 (0.003715) Wilderness_Area2
30. feature 46 (0.003691) Soil_Type33
31. feature 24 (0.003442) Soil_Type11
32. feature 19 (0.002861) Soil_Type6
33. feature 37 (0.002609) Soil_Type24
34. feature 44 (0.002392) Soil_Type31
35. feature 48 (0.002167) Soil_Type35
36. feature 33 (0.001731) Soil_Type20
37. feature 14 (0.001660) Soil_Type1
38. feature 18 (0.001202) Soil_Type5
39. feature 29 (0.001043) Soil_Type16
40. feature 31 (0.000798) Soil_Type18
41. feature 27 (0.000665) Soil_Type14
42. feature 50 (0.000626) Soil_Type37
43. feature 39 (0.000466) Soil_Type26
44. feature 32 (0.000370) Soil_Type19
45. feature 47 (0.000299) Soil_Type34
46. feature 34 (0.000262) Soil_Type21
47. feature 40 (0.000183) Soil_Type27
48. feature 41 (0.000158) Soil_Type28
49. feature 22 (0.000115) Soil_Type9
50. feature 49 (0.000098) Soil_Type36
51. feature 38 (0.000019) Soil_Type25
52. feature 21 (0.000005) Soil_Type8
53. feature 28 (0.000000) Soil_Type15
54. feature 20 (0.000000) Soil_Type7
```

In [50]:
```
sortedImportances = np.flip(np.sort(importances))
```

In [61]:
```
plt.plot(sortedImportances)
plt.savefig('randForImpt.png')
```



In [52]:
```
randFor.score(test_features, test_typeCats)
```

Out[52]: 0.8747795414462081

In [53]:
```
mean_squared_error(test_typeCats, randFor.predict(test_features))
```

Out[53]: 1.0044091710758378

Singular Values

In [54]:
```
from scipy import linalg
```

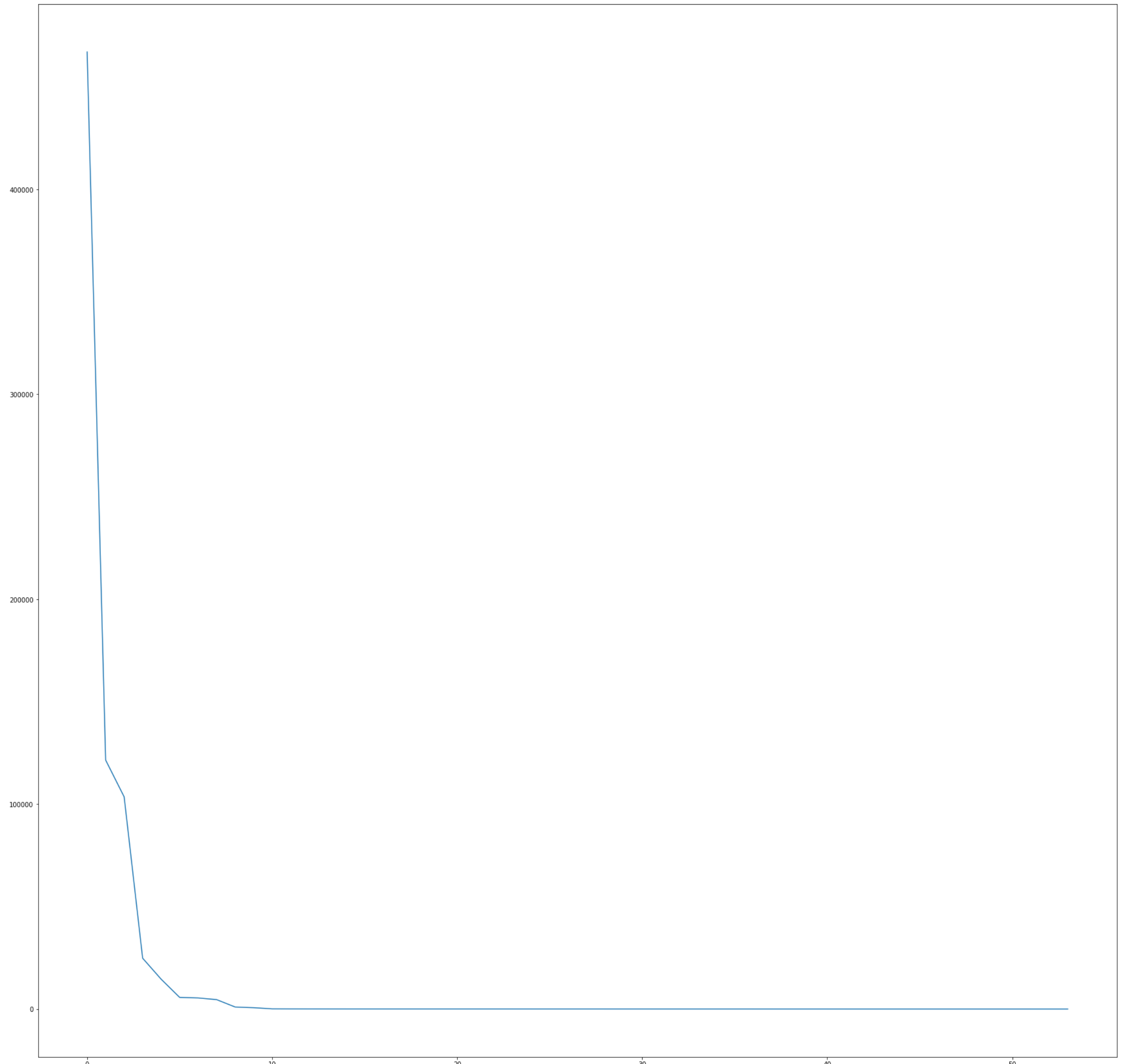
In [55]:
```
U, s, Vh = linalg.svd(features)
```

In [56]:
```
s
```

Out[56]:
```
array([4.67079987e+05, 1.21491986e+05, 1.03540401e+05, 2.47911813e+04,
       1.45494480e+04, 5.66835000e+03, 5.40132217e+03, 4.57376828e+03,
       9.72450727e+02, 6.38460652e+02, 6.89717394e+01, 4.66463768e+01,
       3.66664850e+01, 3.09044312e+01, 2.82551809e+01, 2.70954529e+01,
       2.65842403e+01, 2.59223293e+01, 2.53672078e+01, 2.51578829e+01,
       2.45475462e+01, 2.39910347e+01, 2.28458439e+01, 2.10901401e+01,
       2.04171849e+01, 1.96009908e+01, 1.92450302e+01, 1.87555044e+01,
       1.79981699e+01, 1.70122248e+01, 1.61720007e+01, 1.43085442e+01,
       1.31889974e+01, 1.27983174e+01, 1.18615203e+01, 1.07671856e+01,
       1.05579198e+01, 8.49477246e+00, 7.49978983e+00, 7.19352562e+00,
       6.76847704e+00, 5.65923640e+00, 4.73126542e+00, 4.04604355e+00,
       3.89114589e+00, 3.24002600e+00, 3.12906688e+00, 3.00425301e+00,
       1.05439097e+00, 1.00102947e+00, 9.82337590e-01, 3.41671506e-11,
       3.41671506e-11, 3.41671506e-11])
```

In [57]:
```
sortedS = np.flip(np.sort(s))
```

In [62]:
```
plt.plot(sortedS)
plt.savefig('SVDimport.png')
```



In [65]:
```
U = 0
s = 0
Vh = 0
```

In [ ]:
```
```