Assignment 3 Part 2

CS4172 Machine Learning Lab

Name: Abhiroop Mukherjee

Enrolment Number: 510519109

Task 4

Download the Forest Cover Type dataset (https://www.kaggle.com/uciml/forest-cover-type-dataset) and pre-process the dummy variables to create training, test, and development set. Reduce the train data size if the system unable to process the whole dataset.

In [ ]:
```
import pandas as pd

_FILE_PATH = './../ML_DRIVE/Assign_3/covtype/covtype.csv'

cov_df = pd.read_csv(_FILE_PATH)

cov_df
```

Out[ ]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade_Noo
0	2596	51	3	258	0	510	221	23
1	2590	56	2	212	-6	390	220	23
2	2804	139	9	268	65	3180	234	23
3	2785	155	18	242	118	3090	238	23
4	2595	45	2	153	-1	391	220	23
...	...	...	...	...	...	...	...	.
581007	2396	153	20	85	17	108	240	23
581008	2391	152	19	67	12	95	240	23
581009	2386	159	17	60	7	90	236	24
581010	2384	170	15	60	5	90	230	24
581011	2383	165	13	60	4	67	231	24

581012 rows × 55 columns

In [ ]:

```
cov_df.columns
```

Out[ ]:

```
Index(['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'],
      dtype='object')
```

In [ ]:

```
from sklearn.preprocessing import StandardScaler

def standardize(df: "pd.DataFrame", col_name: "str") -> "pd.DataFrame":
```

```
    scaler = StandardScaler()

    df[[col_name]] = pd.DataFrame(
        data=scaler.fit_transform(df[[col_name]]),
        index=df.index,
        columns=[col_name]
    )
    return df
```

In [ ]:
```
_columns_to_scale = ['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']

for _col in _columns_to_scale:
    cov_df = standardize(cov_df, _col)

cov_df
```

Out[ ]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshad
0	-1.297805	-0.935157	-1.482820	-0.053767	-0.796273	-1.180146	0.330743	0
1	-1.319235	-0.890480	-1.616363	-0.270188	-0.899197	-1.257106	0.293388	0
2	-0.554907	-0.148836	-0.681563	-0.006719	0.318742	0.532212	0.816364	0
3	-0.622768	-0.005869	0.520322	-0.129044	1.227908	0.474492	0.965786	0
4	-1.301377	-0.988770	-1.616363	-0.547771	-0.813427	-1.256464	0.293388	0
...	...	...	...	...	...	...	...	
581007	-2.012130	-0.023740	0.787408	-0.867697	-0.504653	-1.437962	1.040496	0
581008	-2.029988	-0.032675	0.653865	-0.952383	-0.590424	-1.446299	1.040496	0
581009	-2.047847	0.029873	0.386780	-0.985317	-0.676194	-1.449506	0.891075	0
581010	-2.054990	0.128163	0.119694	-0.985317	-0.710502	-1.449506	0.666942	1
581011	-2.058562	0.083486	-0.147392	-0.985317	-0.727656	-1.464256	0.704298	1

581012 rows × 55 columns

In [ ]:
```
cov_df[['Cover_Type']].value_counts()
```

Out[ ]:
```
Cover_Type
2             283301
1             211840
3              35754
7              20510
6              17367
5               9493
4               2747
dtype: int64
```

In [ ]:
```
# NOTE: class imbalance is present but removing it will
# remove the data that cover_type 2 is the most common data in world

cov_df = cov_df.sample(frac=0.1)

X = cov_df.drop('Cover_Type', axis=1)
y = cov_df[['Cover_Type']]
```

In [ ]:
```
y.value_counts()
```

Out[ ]:
```
Cover_Type
2             28425
1             21189
3              3522
7              2110
6              1672
5               911
4               272
dtype: int64
```

In [ ]:
```
# 80% as train
# 10% as validation
# 10% as train

from sklearn.model_selection import train_test_split

X_train, _X_rest, y_train, _y_rest = train_test_split(X, y, train_size=0.8)
X_val, X_test, y_val, y_test = train_test_split(_X_rest, _y_rest, train_size=0.5)
```

Task 5

Apply multi-class classification in SVM using Forest Cover Type dataset.

In [ ]:
```
# https://scikit-learn.org/stable/modules/svm.html#svm
```

```python
# chose LinearSVC cause no mention of kernel to be used
# and LinearSVC is the fastest


# https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, f1_score

_model = LinearSVC(max_iter=1000000).fit(X_train, y_train.iloc[:, 0])

_y_predict = _model.predict(X_val)
_accuracy = accuracy_score(y_val, _y_predict)
# 'weighted' cause it also takes imbalance of classes into account
_f1 = f1_score(y_val, _y_predict, average='weighted')

print(f"default accuracy = {_accuracy}")
print(f"default f1 = {_f1}")
```

```
default accuracy = 0.7189328743545611
default f1 = 0.7027670149054814
```

In [ ]:

```python
# hyper parameter tuning

def svm_train(
    X_train: "pd.DataFrame",
    X_val: "pd.DataFrame",
    y_train: "pd.DataFrame",
    y_val: "pd.DataFrame",
    tol=1e-4,
    C=1.0
) -> "LinearSVC":
    '''
    Wrapper Function for sklearn.svm.LinearSVC
    See: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
    '''

    model = LinearSVC(
        tol=tol,
        C=C,
        max_iter=1000000
    ).fit(X_train, y_train.iloc[:, 0])

    y_predict = model.predict(X_val)
    accuracy = accuracy_score(y_val, y_predict)
    f1 = f1_score(y_val, y_predict, average='weighted')

    return [tol, C, accuracy, f1]
```

In [ ]:
```
# Test 1: tolerance test
_result = []
_tolerances = [1e-5, 3.3333e-5, 6.6666e-5, 1e-4,
               3.333e-4, 6.666e-4, 1e-3]

for tol in _tolerances:
    _result.append(svm_train(X_train, X_val, y_train, y_val, tol=tol))

_df = pd.DataFrame(
    data=_result,
    columns=["Tolerance", "C", "Accuracy", "f1"]
)
print(_df)
# also sort by ascending Tolerance as less tolerance means
# less training time
best_tol = _df.sort_values(
    ['f1', 'Accuracy', 'Tolerance'], ascending=False
).iloc[0, :]['Tolerance']
print(f"best tolerance = {best_tol}")
```

```
   Tolerance    C  Accuracy        f1
0   0.000010  1.0  0.718933  0.702767
1   0.000033  1.0  0.718933  0.702767
2   0.000067  1.0  0.718933  0.702767
3   0.000100  1.0  0.718933  0.702767
4   0.000333  1.0  0.718933  0.702767
5   0.000667  1.0  0.718933  0.702767
6   0.001000  1.0  0.718933  0.702767
best tolerance = 0.001
```

In [ ]:
```
# Test 2: C values

_result = []
_c_values = [0.3333, 0.6666, 1, 3.3333, 6.6666, 10]

for c in _c_values:
    _result.append(svm_train(X_train, X_val, y_train, y_val,
                             tol=best_tol, C=c))

_df = pd.DataFrame(
    data=_result,
    columns=["Tolerance", "C", "Accuracy", "f1"]
)

print(_df)
```

```
# also sort by ascending C as for same result, smallest C
# means biggest 1/C which means stronger regularization
best_C = _df.sort_values(
    ['f1', 'Accuracy', 'C'], ascending=[False, False, True]
).iloc[0, :]['C']
print(f"best tolerance = {best_C}")
```

```
   Tolerance        C  Accuracy        f1
0      0.001   0.3333  0.718417  0.702008
1      0.001   0.6666  0.719105  0.702915
2      0.001   1.0000  0.718933  0.702767
3      0.001   3.3333  0.718761  0.702662
4      0.001   6.6666  0.718933  0.703002
5      0.001  10.0000  0.718933  0.703002
best tolerance = 6.6666
```

In [ ]:

```
# End Result after using validation dataset for all hyper-parameters

model = LinearSVC(
        tol=best_tol,
        C=best_C,
        max_iter=1000000
    ).fit(X_train, y_train.iloc[:, 0])

y_predict = model.predict(X_test)
accuracy = accuracy_score(y_test, y_predict)
f1 = f1_score(y_test, y_predict, average='weighted')

print(f"Test Accuracy: {accuracy}")
print(f"Test F1: {f1}")
```

```
Test Accuracy: 0.7184649802099466
Test F1: 0.7029568428478489
```

Task 6

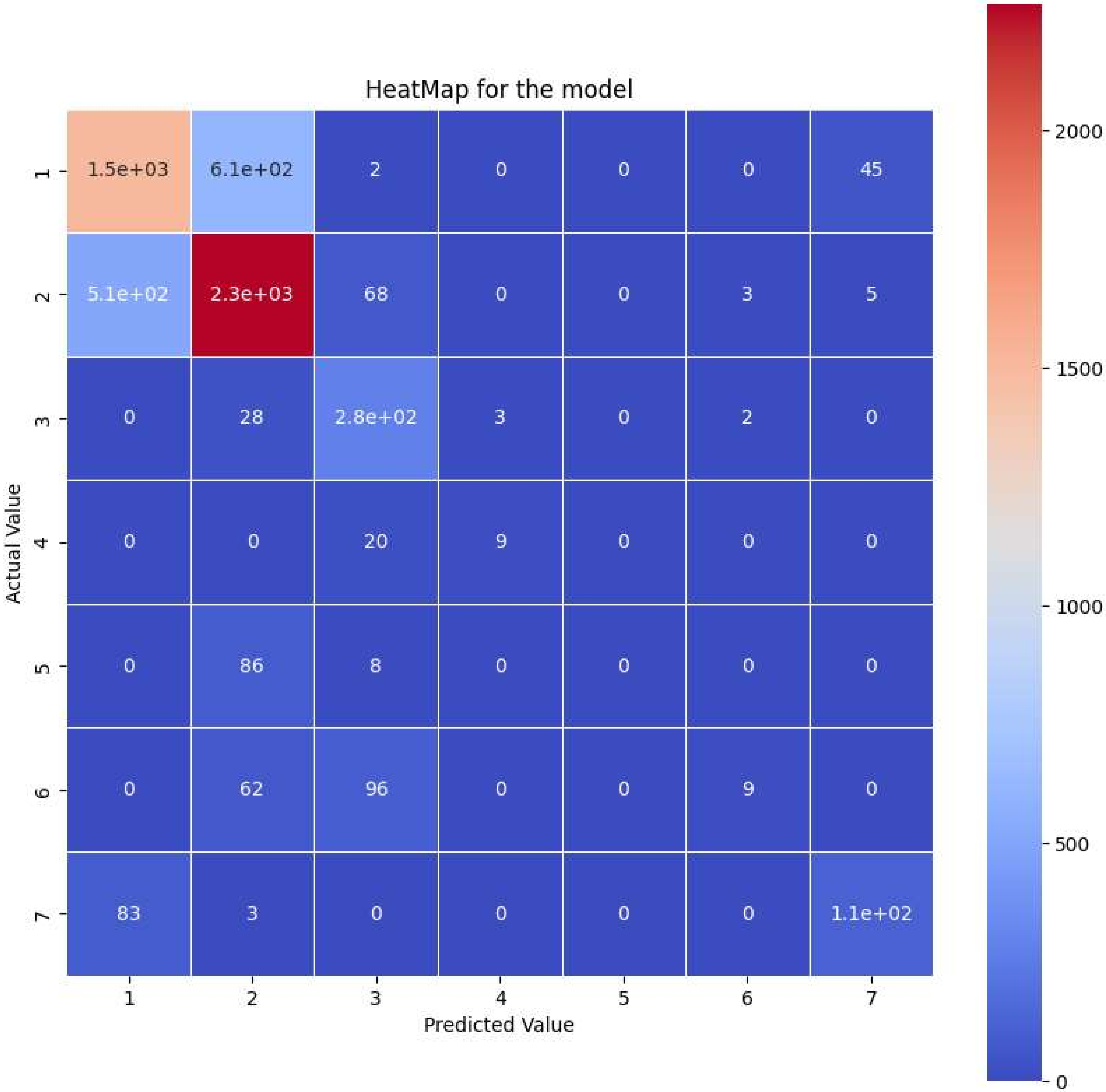
Plot and Analyze the Confusion matrix for the above applied SVM method.

In [ ]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

matrix = confusion_matrix(y_test, y_predict)
fig = plt.figure(figsize=(10,10))
sns.heatmap(
```

```
    matrix,
    xticklabels=range(1,8),
    yticklabels=range(1,8),
    linewidth=0.5,
    cmap='coolwarm',
    annot=True,
    cbar=True,
    square=True)
plt.title('HeatMap for the model')
plt.ylabel('Actual Value')
plt.xlabel('Predicted Value')
plt.show()
```


HeatMap for the model
Actual Value
Predicted Value
1.5e+03
6.1e+02
2
0
0
0
45
5.1e+02
2.3e+03
68
0
0
3
5
0
28
2.8e+02
3
0
2
0
0
0
20
9
0
0
0
0
86
8
0
0
0
0
0
62
96
0
0
9
0
83
3
0
0
0
0
1.1e+02
1
2
3
4
5
6
7
2000
1500
1000
500
0

Task 7

Consider only two features and three classes and train Logistic Regression 3-class Classifier (Any three-class) to show the training and test area in a 2-D plane, using matplotlib.

In [ ]:
```
X_train
```

Out[ ]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshad
16722	-0.262033	-1.104931	-0.681563	0.341435	-0.350267	1.862336	0.218677	-0
166952	0.805883	0.458775	0.386780	1.244756	1.073522	2.308703	-0.192233	1
283277	0.759452	-1.069189	0.253237	2.533869	-1.671130	-0.930668	0.181321	-1
552577	1.173760	0.405162	-0.414477	-0.467789	-0.744810	-0.296390	0.143966	1
63762	0.666589	-0.863673	-0.147392	-0.321941	-0.556116	0.920218	0.666942	-0
...	...	...	...	...	...	...	...	
488243	1.063040	0.038808	0.253237	0.477874	0.387359	-0.846012	0.853719	0
481639	-0.344181	-0.881544	0.253237	-0.952383	-0.573270	-1.044183	0.629587	-0
179037	-0.244175	-0.997705	0.253237	0.073262	-1.413819	-1.076891	0.368099	-1
330017	0.363001	-1.104931	-0.147392	0.035624	0.455975	-0.089881	0.143966	-0
476325	0.180848	0.503452	0.787408	0.426122	-0.984967	-0.754942	-0.491076	1

46480 rows × 54 columns

In [ ]:
```
#  taking first two features

subset_X_train = X_train.iloc[:, 0:2]
subset_y_train = y_train

subset_train = subset_X_train.join(subset_y_train)
subset_train = subset_train[subset_train['Cover_Type'].isin([1,2,3])]

subset_train
```

Out[ ]:

	Elevation	Aspect	Cover_Type
16722	-0.262033	-1.104931	2
166952	0.805883	0.458775	2
283277	0.759452	-1.069189	2
552577	1.173760	0.405162	1
63762	0.666589	-0.863673	1
...	...	...	...
488243	1.063040	0.038808	2
481639	-0.344181	-0.881544	2
179037	-0.244175	-0.997705	2
330017	0.363001	-1.104931	1
476325	0.180848	0.503452	2

42511 rows × 3 columns

In [ ]:

```
model = LinearSVC().fit(
    subset_train.iloc[:, 0:2],
    subset_train.iloc[:, 2]
)
```

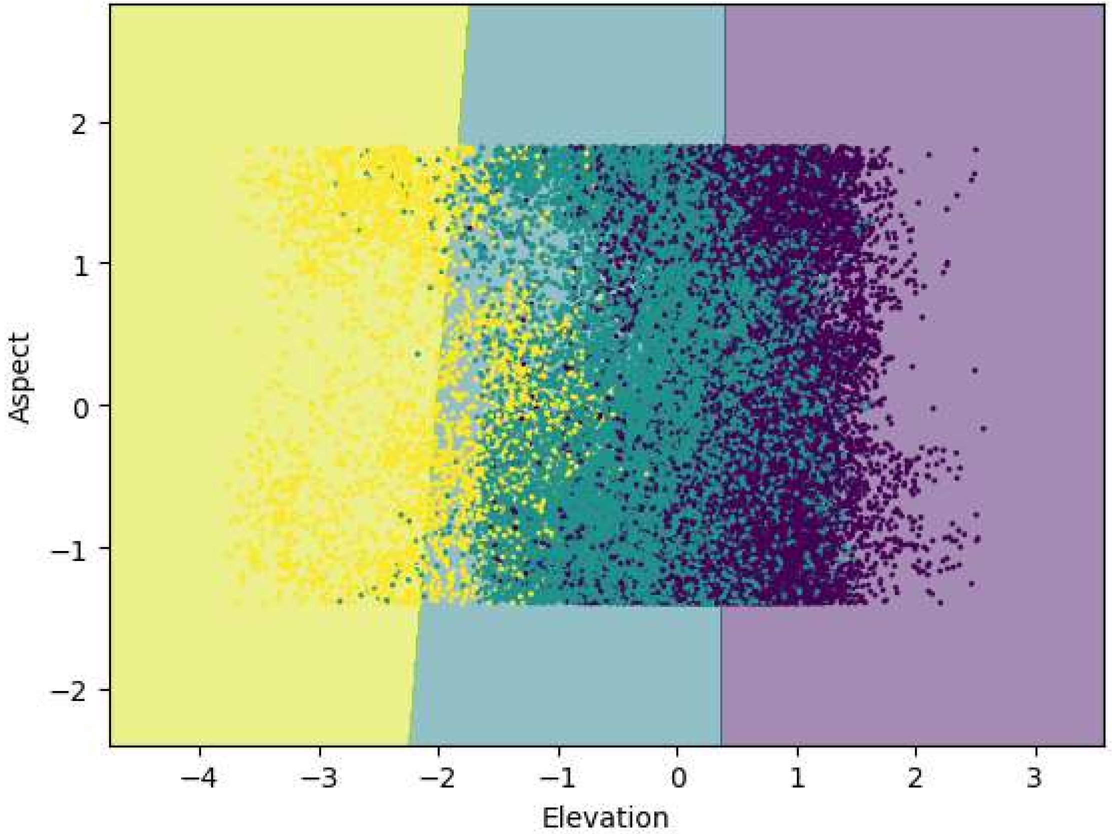
In [ ]:

```
from sklearn.inspection import DecisionBoundaryDisplay

disp = DecisionBoundaryDisplay.from_estimator(
    model,
    subset_train.iloc[:, 0:2],
    xlabel='Elevation',
    ylabel='Aspect',
    alpha=0.5,
    grid_resolution=5000
)

disp.ax_.scatter(
    subset_train['Elevation'],
    subset_train['Aspect'],
    c=subset_train['Cover_Type'],
    s=1
)
```

Out[ ]: <matplotlib.collections.PathCollection at 0x1438b2c0100>



In [ ]:
```
subset_X_test = X_test.iloc[:, 0:2]
subset_y_test = y_test

subset_test = subset_X_test.join(subset_y_test)
subset_test = subset_test[subset_test['Cover_Type'].isin([1,2,3])]

disp = DecisionBoundaryDisplay.from_estimator(
    model,
    subset_test.iloc[:, 0:2],
    xlabel='Elevation',
    ylabel='Aspect',
    alpha=0.5,
    grid_resolution=5000
)

disp.ax_.scatter(
    subset_test['Elevation'],
    subset_test['Aspect'],
```

```python
    c=subset_test['Cover_Type'],
    s=3
)
```

Out[ ]: <matplotlib.collections.PathCollection at 0x1438b34e740>