FOREST COVER TYPE

Name : Praveen Kumar S
Reg.No : 11809946
Roll No : A22

Department of computer science and engineering, Lovely ProfessionalUniversity,Phagwara,India

ABSTRACT

The study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. Each observation is a 30m x 30m patch. You are asked to predict an integer classification for the forest cover type. The seven types are:

1 - Spruce/Fir

2 - Lodgepole Pine

3 - Ponderosa Pine

4 - Cottonwood/Willow

5 - Aspen

6 - Douglas-fir

7 - Krummholz

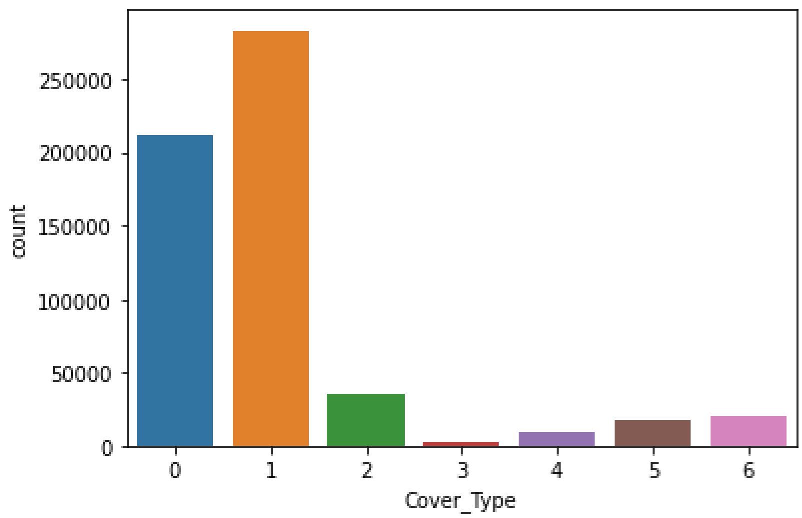
The training set (15120 observations) contains both features and the Cover_Type. The test set contains only the features. You must predict the Cover_Type for every row in the test set (565892 observations).

INTRODUCTION

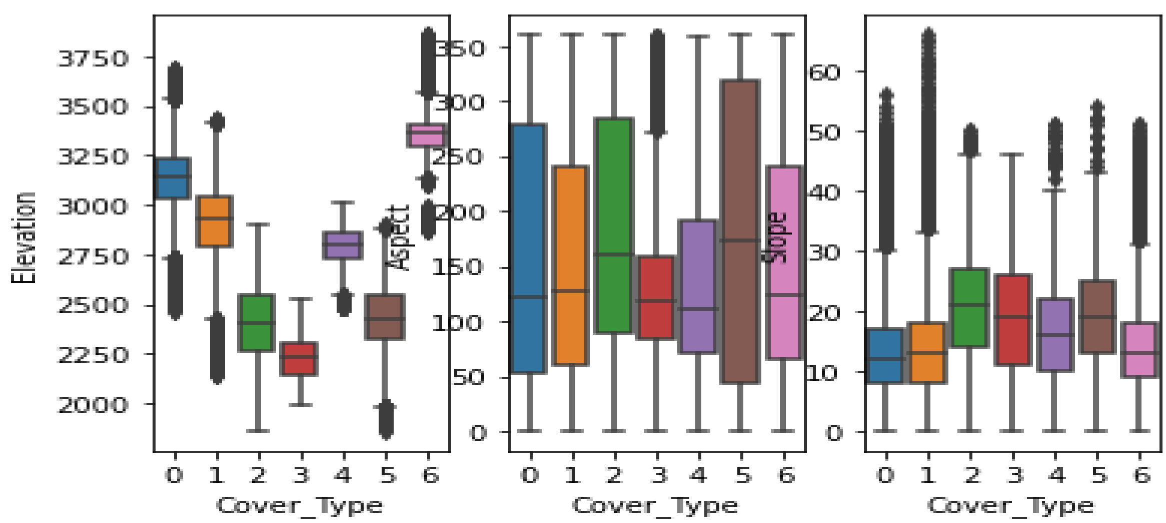
For this assignment I have chosen a kaggle competition ’For-est Cover Type Prediction’[1]. This data is obtained from theUS Geological Survey (USGS) and the US Forest Service(USFS) and includes four wilderness areas located in Roo-sevelt National Forest of northern Colorado, and provided by Machine Learning Laboratory of University of CaliforniaIrvine[1]. Dataset contains 581k entries with 54 attributes each. However, there are only 12 real features because two of them are represented as a vector opposed to number no-tation. Each entry is observation on 30×30 m patch of forest land and goal of the competition is to predict cover type of thispatch. Training set is chosen in so fashion that each class has the same number of observations.

Explanatory Data Analysis

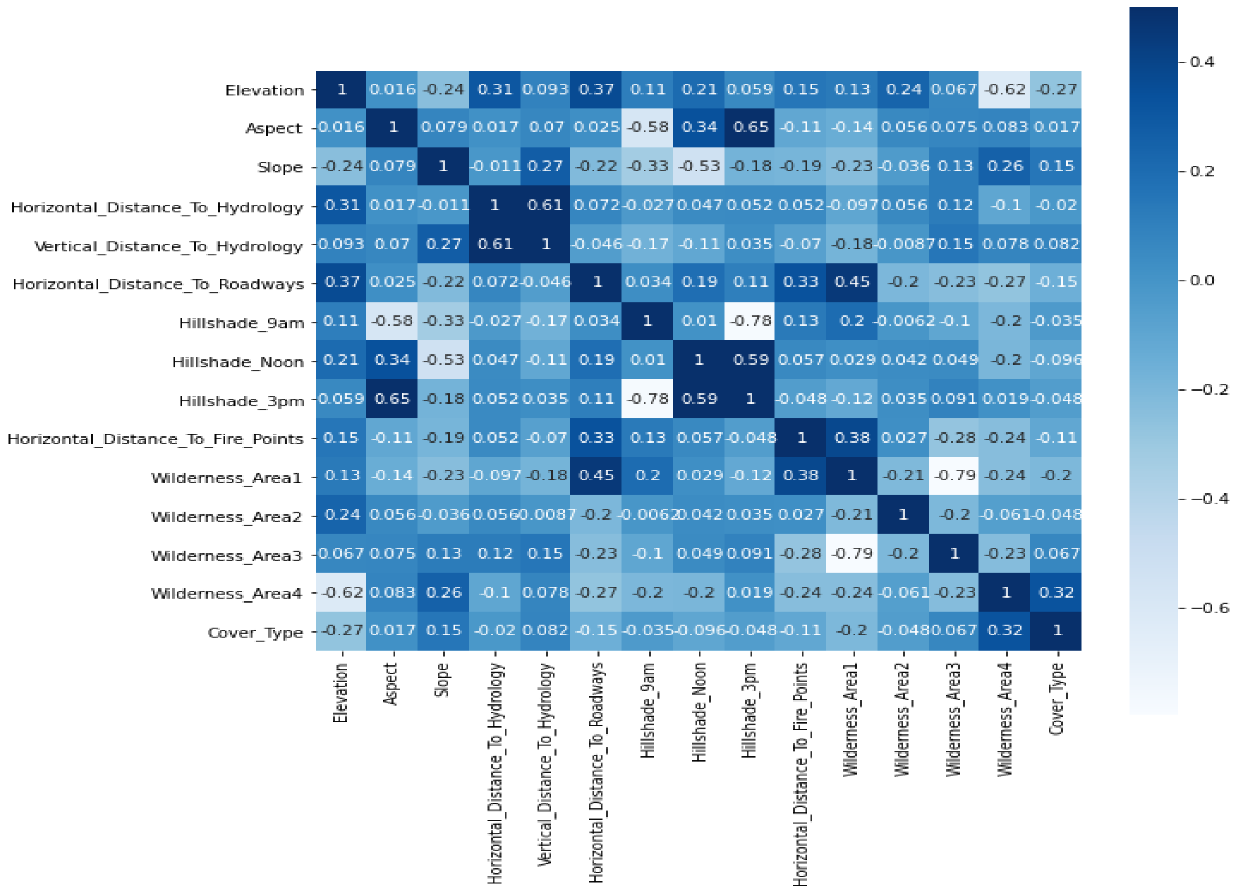
EDA is the first step in this workflow where the decision-making process is initiated for the feature selection. Some valuable insights can be obtained by looking at the distribution of the target, relationship of the features to the target and link between the features. My preference is to start by looking at the target, then examine the features and its relations to the target



Box plot to check how well the data distribution is and it helped to identify the outliers in the data.



The target values were plotted to check its distribution and below i have checked the corelation between the features of data.



PREPROCESSING :

STANDADIZATION : Data has been standadized as the dataset has different scaling and use of distance based algorithm would affect them .So i standard scaled the data using Standard scaler using library from skit-learn.

Standadization

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

```
[ ] scaler=StandardScaler()
    x=pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
```

Principle Component Analysis: Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

Principal Component Analysis

```
[ ] from sklearn.decomposition import PCA
    pca = PCA(n_components=13)
    principle=pca.fit_transform(x)
```

```
[ ] x=pd.DataFrame(data=principle,columns=['pca1','pca2','pca3','pca4','pca5','pca6','pca7','pca8','pca9','pca10','pca11','pca12','pca13'])
    x.head()
```

	pca1	pca2	pca3	pca4	pca5	pca6	pca7	pca8	pca9	pca10	pca11	pca12	pca13
0	-2.436748	1.273240	-0.846860	-0.282383	-0.749459	0.630397	0.514132	-0.932146	1.067557	-0.259989	0.106124	0.224373	0.211067
1	-2.449171	1.224911	-1.017107	-0.494141	-0.746207	0.584671	0.507774	-0.923433	1.068863	-0.264661	0.116323	0.230402	0.217273
2	-2.726381	0.746456	-0.608186	0.108083	-1.324557	1.672583	3.249161	-0.097704	-0.557931	-0.006895	-0.386931	-1.001769	-0.823864
3	-2.108939	1.170434	0.004021	1.286132	-0.814300	0.114126	1.249211	2.295227	-1.359482	0.665203	0.872558	0.840057	1.074424
4	-2.443913	1.320752	-0.992714	-0.556693	-0.728546	0.491427	0.491026	-0.924997	1.054773	-0.268491	0.119525	0.223050	0.217175

IMPLEMENTATION OF MODELS

I used 3 models in this code so,the three models are

Model 1 : Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.So the prerequisites for random forest to perform well are:

1.There needs to be some actual signal in our features so that models built using those features do better than random guessing.

2.The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Model 2 : The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.

Model 3: A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

Result and discussion:

Step 1: This is the program which i have implemented in the program importing all the important libraries like numpy, pandas, seaborn and machine learning algorithms.

Importing Library

```
[ ] import numpy as np
    import pandas as pd
    import warnings
    import matplotlib.pyplot as plt
    import seaborn as sns

    from sklearn.decomposition import PCA
    from sklearn.preprocessing import normalize
    from sklearn.manifold import TSNE

    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import accuracy_score,log_loss
    from sklearn.metrics import plot_confusion_matrix

    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.svm import SVC
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.calibration import CalibratedClassifierCV
    from sklearn import svm

    from sklearn.model_selection import train_test_split
    from sklearn.model_selection import GridSearchCV

    import math

    warnings.filterwarnings("ignore")
```

Step 2 :: After that i am read.csv file and i will get output these are data values which consists of 581012 rows and 54 columns.

Reading Forest Cover Data

```
[ ] data=pd.read_csv('/content/drive/MyDrive/Kaggle/covtype.csv')
    print("number of datapoints",data.shape[0])
    print("number of features",data.shape[1])
    data.head()
```

```
number of datapoints 581012
number of features 55
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade_Noon	Hillshade_3
0	2596	51	3	258	0	510	221	232	1
1	2590	56	2	212	-6	390	220	235	1
2	2804	139	9	268	65	3180	234	238	1
3	2785	155	18	242	118	3090	238	238	1
4	2595	45	2	153	-1	391	220	234	1

Step 3 : After that i am describing the dataset like count ,mean of the rows,std,min,25percent,50 percent,75percent max using the describe() (describe function).

```
[ ] data.describe()
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade
count	581012.000000	581012.000000	581012.000000	581012.000000	581012.000000	581012.000000	581012.000000	581012.
mean	2959.365301	155.656807	14.103704	269.428217	46.418855	2350.146611	212.146049	223.
std	279.984734	111.913721	7.488242	212.549356	58.295232	1559.254870	26.769889	19.
min	1859.000000	0.000000	0.000000	0.000000	-173.000000	0.000000	0.000000	0.
25%	2809.000000	58.000000	9.000000	108.000000	7.000000	1106.000000	198.000000	213.
50%	2996.000000	127.000000	13.000000	218.000000	30.000000	1997.000000	218.000000	226.
75%	3163.000000	260.000000	18.000000	384.000000	69.000000	3328.000000	231.000000	237.
max	3858.000000	360.000000	66.000000	1397.000000	601.000000	7117.000000	254.000000	254.

Step 4 : After that this data has interpreting the values as the null values which as the null value are not.

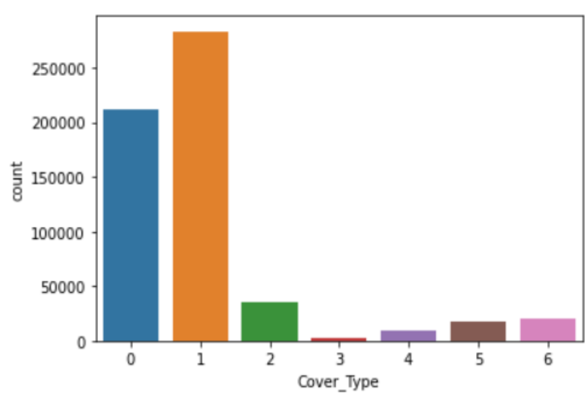
```
[ ] data[data.isnull().any(axis=1)]

     Elevation  Aspect  Slope  Horizontal_Distance_To_Hydrology  Vertical_Distance_To_Hydrology  Horizontal_Distance_To_Roadways  Hillshade_9am  Hillshad
```

```
[ ] data.isnull().sum()

    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
```

Step 5 : After that i plotted a countplot to look at the distribution of target values.

```
[ ] data.Cover_Type.unique()

    array([4, 1, 0, 6, 2, 5, 3])
```

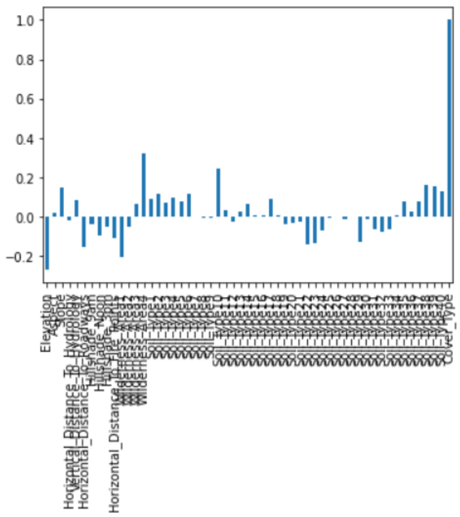
```
[ ] ax = sns.countplot(x="Cover_Type", data=data)
```



Step6 : As we have analysed the target values let us see at correlation of other features with respect to target data.

```
[ ] data.corr()["Cover_Type"].plot(kind="bar")
```

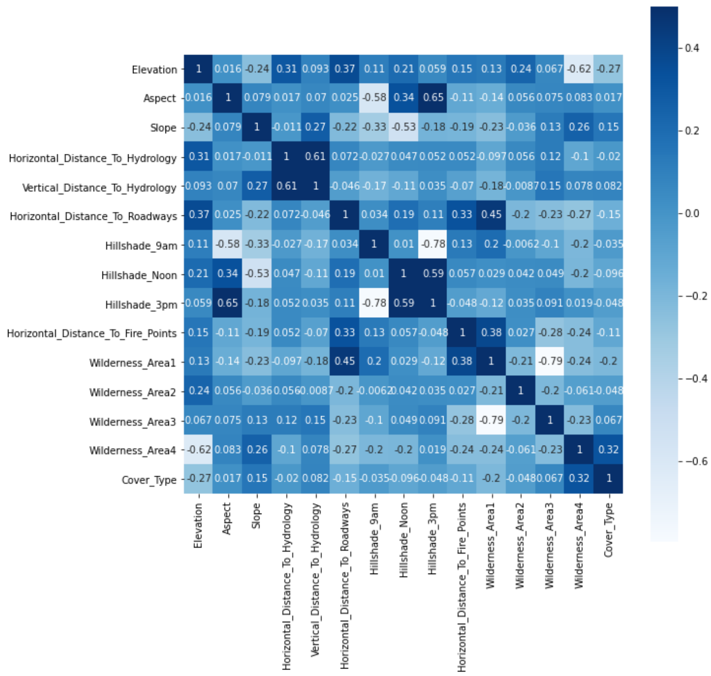
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8feb2ca410>
```



Step 7 : Used a heat map to analyse how the features are co-related with each other.

```
fig = plt.subplots(figsize=(10,10))
sns.heatmap(df1.corr(),vmax=0.5,square=True,annot=True,cmap='Blues')
plt.xticks(rotation=90)
plt.yticks(rotation=0)
```

```
(array([ 0.5,  1.5,  2.5,  3.5,  4.5,  5.5,  6.5,  7.5,  8.5,  9.5, 10.5,
        11.5, 12.5, 13.5, 14.5]), <a list of 15 Text major ticklabel objects>)
```



Step 8 : After that i implemented a Random forest classifier with hyperparameter tuning using calibrated cv and calculated log loss for each hyperparameter and choosed the best hyperparameter to work on.

```
alpha = [100,200]
max_depth = [5, 10]
cv_log_error_array = []
for i in alpha:
    for j in max_depth:
        print("for n_estimators =", i,"and max depth = ", j)
        clf = RandomForestClassifier(n_estimators=i, criterion='gini', max_depth=j, random_state=42, n_jobs=-1)
        clf.fit(train_df,y_train)
        sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
        sig_clf.fit(train_df,y_train)
        sig_clf_probs = sig_clf.predict_proba(cv_df)
        cv_log_error_array.append(log_loss(y_cv, sig_clf_probs, labels=clf.classes_, eps=1e-15))
        print("Log Loss :",log_loss(y_cv, sig_clf_probs))


best_alpha = np.argmin(cv_log_error_array)
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_depth=max_depth[int(best_alpha%2)], random_state=42, n_jobs=-1)
clf.fit(train_df, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_df,y_train)

predict_y = sig_clf.predict_proba(train_df)
print('For values of best estimator = ', alpha[int(best_alpha/2)], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_df)
print('For values of best estimator = ', alpha[int(best_alpha/2)], "The cross validation log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best estimator = ', alpha[int(best_alpha/2)], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

```
for n_estimators = 100 and max depth =  5
Log Loss : 0.7155034757634388
for n_estimators = 100 and max depth =  10
Log Loss : 0.5708224909882673
for n_estimators = 200 and max depth =  5
Log Loss : 0.7139458001310026
for n_estimators = 200 and max depth =  10
Log Loss : 0.5718205689222435
For values of best estimator =  100 The train log loss is: 0.5631860465615696
For values of best estimator =  100 The cross validation log loss is: 0.5708224909885047
For values of best estimator =  100 The test log loss is: 0.5754977818664438
```

After choosing the best parameter i created a model and plotted confusion matrix.

Training and Testing the model with best hyperparameter - RF

```
[ ] clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_depth=max_depth[int(best_alpha%2)], random_state=42, n_jobs=-1)
    clf.fit(train_df,y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_df,y_train)
    pred_y = sig_clf.predict(X_test)
```

```
[ ] accuracy_score(y_test, pred_y)
```

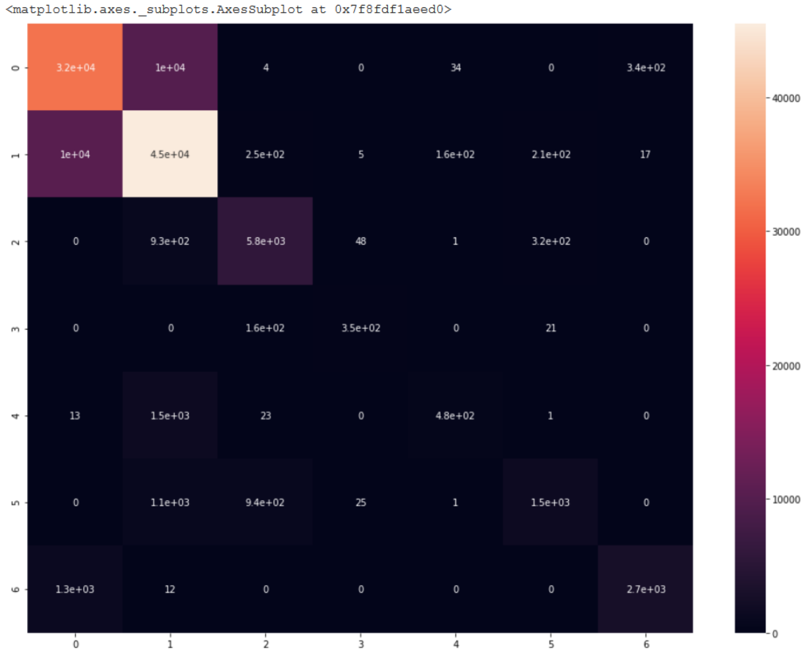
```
0.7605225338416391
```

```
[ ] cf_matrix=confusion_matrix(y_test,pred_y)
    print(cf_matrix)
```

```
[[32081 10099     4     0    34     0   339]
 [10406 45457   246     5   156   213    17]
 [    0   928  5823    48     1   321     0]
 [    0     0   157   348     0    21     0]
 [   13  1477    23     0   481     1     0]
 [    0  1060   935    25     1  1468     0]
 [ 1286    12     0     0     0     0  2717]]
```

```
[ ] f, ax = plt.subplots(figsize=(16, 12))
    sns.heatmap(cf_matrix,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8fdf1aeed0>
```



Step 11 : Now i implemented some other model to check which model performs the best. I planned to use distance based algorithm i preprocessed the data with standard scaling and principle component analysis.

Standadization

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

```
scaler=StandardScaler()
x=pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
```

Principal Component Analysis

```
from sklearn.decomposition import PCA
pca = PCA(n_components=13)
principle=pca.fit_transform(x)
```

```
x=pd.DataFrame(data=principle,columns=['pca1','pca2','pca3','pca4','pca5','pca6','pca7','pca8','pca9','pca10','pca11','pca12','pca13'])
x.head()
```

	pca1	pca2	pca3	pca4	pca5	pca6	pca7	pca8	pca9	pca10	pca11	pca12	pca13
0	-2.436748	1.273240	-0.846860	-0.282383	-0.749459	0.630397	0.514132	-0.932146	1.067557	-0.259989	0.106124	0.224373	0.211067
1	-2.449171	1.224911	-1.017107	-0.494141	-0.746207	0.584671	0.507774	-0.923433	1.068863	-0.264661	0.116323	0.230402	0.217273
2	-2.726381	0.746456	-0.608186	0.108083	-1.324557	1.672583	3.249161	-0.097704	-0.557931	-0.006895	-0.386931	-1.001769	-0.823864
3	-2.108939	1.170434	0.004021	1.286132	-0.814300	0.114126	1.249211	2.295227	-1.359482	0.665203	0.872558	0.840057	1.074424
4	-2.443913	1.320752	-0.992714	-0.556693	-0.728546	0.491427	0.491026	-0.924997	1.054773	-0.268491	0.119525	0.223050	0.217175

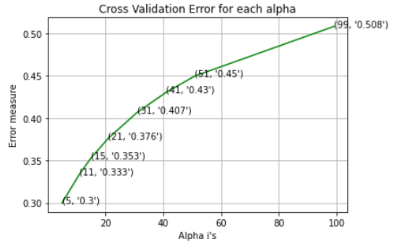
Train Test and Cross Validation

```
X_train, X_test, y_train, y_test = train_test_split(x, y,test_size=0.2,random_state=42)
train_df, cv_df, y_train, y_cv = train_test_split(X_train, y_train, stratify=y_train, test_size=0.2)
```

Step 11 : Now i implemented a KNearest Neighbors model with some parameter and calculated the log loss at each parameter tuning and checked how well the model work and vizulalized some output received from the model.

```python
alpha = [5, 11, 15, 21, 31, 41, 51, 99]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = KNeighborsClassifier(n_neighbors=i)
    clf.fit(train_df,y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_df,y_train)
    sig_clf_probs = sig_clf.predict_proba(cv_df)
    cv_log_error_array.append(log_loss(y_cv, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    # to avoid rounding error while multiplying probabilites we use log-probability estimates
    print("Log Loss :",log_loss(y_cv, sig_clf_probs))

fig, ax = plt.subplots()
ax.plot(alpha, cv_log_error_array,c='g')
for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[i],str(txt)), (alpha[i],cv_log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(cv_log_error_array)
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
clf.fit(train_df, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_df,y_train)

predict_y = sig_clf.predict_proba(train_df)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_df)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

```
for alpha = 5
Log Loss : 0.2995733470137276
for alpha = 11
Log Loss : 0.33336107955825667
for alpha = 15
Log Loss : 0.35264297432531605
for alpha = 21
Log Loss : 0.3760900882946041
for alpha = 31
Log Loss : 0.4067607854579987
for alpha = 41
Log Loss : 0.43045717146911855
for alpha = 51
Log Loss : 0.44986248498675463
for alpha = 99
Log Loss : 0.5083553297818615
```




```
For values of best alpha =  5 The train log loss is: 0.2078252273110488
For values of best alpha =  5 The cross validation log loss is: 0.2995733470137276
For values of best alpha =  5 The test log loss is: 0.3012559789985328
```

Ater hypertuned the model i used the best parameter to fit the model and used confusion matrix as matrix to display the result.

Training and Testing the model with best hyper paramters -KNN

```
[ ] clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
    clf.fit(train_df,y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_df,y_train)
    pred_y = sig_clf.predict(X_test)
```

```
[ ] accuracy_score(y_test, pred_y)
```

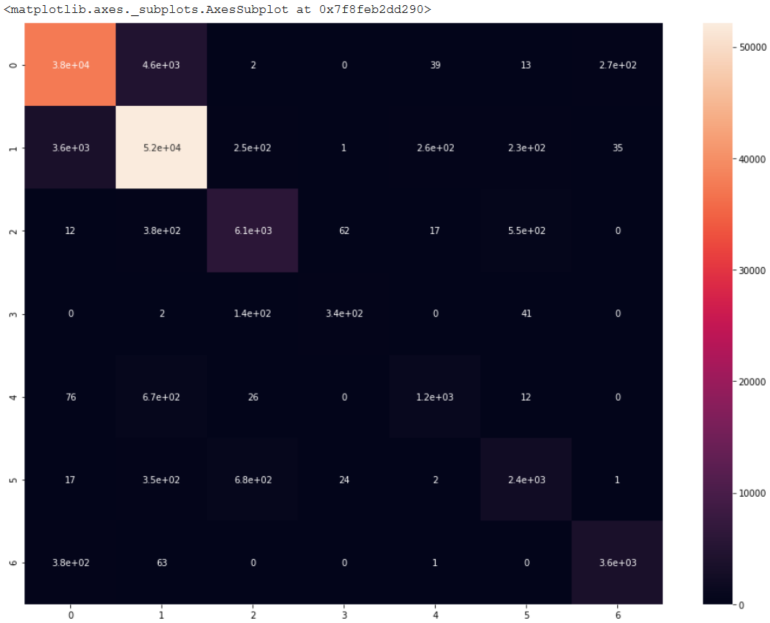
```
0.889486502069654
```

```
[ ] cf_matrix=confusion_matrix(y_test,pred_y)
    print(cf_matrix)
```

```
[[37619  4618     2     0    39    13   266]
 [ 3636 52093   247     1   261   227    35]
 [   12   378  6103    62    17   549     0]
 [    0     2   138   345     0    41     0]
 [   76   669    26     0  1212    12     0]
 [   17   349   683    24     2  2413     1]
 [  375    63     0     0     1     0  3576]]
```

```
[ ] f, ax = plt.subplots(figsize=(16, 12))
    sns.heatmap(cf_matrix,annot=True)
```

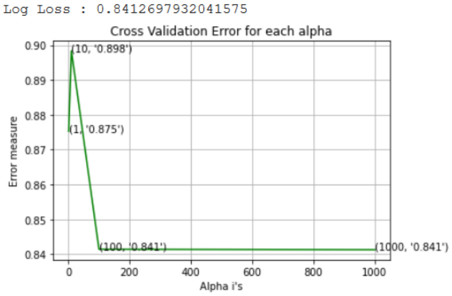
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8feb2dd290>
```



Step 13 : And the i used SVM to see how well my model perform, I used classifiedCV for hyperparameter tun

```
[ ] from sklearn.linear_model import SGDClassifier
    alpha = [1, 10, 100, 1000]
    cv_log_error_array = []
    for i in alpha:
        print("for C =", i)
        clf = SGDClassifier( class_weight='balanced', alpha=i, penalty='l2', loss='hinge', random_state=42)
        clf.fit(train_df,y_train)
        sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
        sig_clf.fit(train_df,y_train)
        sig_clf_probs = sig_clf.predict_proba(cv_df)
        cv_log_error_array.append(log_loss(y_cv, sig_clf_probs, labels=clf.classes_, eps=1e-15))
        print("Log Loss :",log_loss(y_cv, sig_clf_probs))

    fig, ax = plt.subplots()
    ax.plot(alpha, cv_log_error_array,c='g')
    for i, txt in enumerate(np.round(cv_log_error_array,3)):
        ax.annotate((alpha[i],str(txt)), (alpha[i],cv_log_error_array[i]))
    plt.grid()
    plt.title("Cross Validation Error for each alpha")
    plt.xlabel("Alpha i's")
    plt.ylabel("Error measure")
    plt.show()


    best_alpha = np.argmin(cv_log_error_array)
    clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='hinge', random_state=42)
    clf.fit(train_df, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_df,y_train)

    predict_y = sig_clf.predict_proba(train_df)
    print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
    predict_y = sig_clf.predict_proba(cv_df)
    print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
    predict_y = sig_clf.predict_proba(X_test)
    print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

```
[ ]
    for C = 1
    Log Loss : 0.8752553776935269
    for C = 10
    Log Loss : 0.8983556601838384
    for C = 100
    Log Loss : 0.8414010921180827
    for C = 1000
    Log Loss : 0.8412697932041575
```




```
    For values of best alpha =  1000 The train log loss is: 0.8429689221291818
    For values of best alpha =  1000 The cross validation log loss is: 0.8412697932041575
    For values of best alpha =  1000 The test log loss is: 0.8461127254717933
```

After finding the best hyperparameter i used the best hyperparameter and trained a model and used confusion matrix as metrics.

Training and testing the model with best hyperparameter - SVM

```
[ ] clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='hi
    clf.fit(train_df,y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_df,y_train)
    pred_y = sig_clf.predict(X_test)
```

```
[ ] accuracy_score(y_test, pred_y)
```

```
0.6309389602678072
```

```
[ ] cf_matrix=confusion_matrix(y_test,pred_y)
    print(cf_matrix)
```

```
[[19537 22781     0     0     0     0   239]
 [ 6422 49198   659     0     2     0   219]
 [    0  2880  4241     0     0     0     0]
 [    0    11   515     0     0     0     0]
 [   29  1963     0     0     0     0     3]
 [    0  1591  1897     0     1     0     0]
 [ 3174   500     0     0     0     0   341]]
```

```
[ ] f, ax = plt.subplots(figsize=(16, 12))
    sns.heatmap(cf_matrix,annot=True)
```

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7fd2dcf39890&gt;
```

CONCLUSION

The Scope of this project is to apply machine learning concepts taught in class on our data set. We made use of open dataset. We are able to apply several classification machine learning methods on the computer CoverType data set to achieve the goal of the project.The goal of the project is to model the data, so as to accurately predict the type of covertype . Also, infer the relationship between predictors and the response. We achieved these goals by using several classification methods and in this project we have download data sets in the UCI and we have used the data sets in the form of .csv file in this.

REFERENCES

UCI machine learning repository
https://archive.ics.uci.edu/ml/datasets/covertype

Medium article
https://medium.com/cuny-csi-mth513/whats-that-forest-cover-type-c801526d3c18

Project Colab Link

Colab File name : 11809946 Forest Cover Type
Link
https://colab.research.google.com/drive/1mJkXPKtJAFAfS7Z_nFQWT7mgsoQwON5X?usp=sharing