report

November 12, 2018

1 Project: *Project Name*

1.1 *Type of ML project*

1.2 Table of Contents

  * Section 1.8.2
- Section 1.8.3
- Section 1.8.4

- Section 1.10
  - Section 1.10.1
  - Section 1.10.2
  - Section 1.10.3
  - Section 1.10.4
  - Section 1.10.5
  - Section 1.10.6
- Section 1.12
- Section 1.14
-

1.3 Section 1.16

1.4 Getting Started

Giving necessory, important and short info on this project.

1.4.1 Version Check-In

```
In [ ]: # Importing required libraries for the project
        import sys # for python library version
        import numpy as np # for scientific computing
        import pandas as pd # for data anaysis
        import matplotlib # for visualization
        import seaborn as sns # for visualization
        import sklearn # ML Library
        import tensorflow as tf # deep learning framework
```

```
In [ ]: print('Python: {}'.format(sys.version))  # Python version
        print('numpy: {}'.format(np.__version__))  # Numpy version
        print('pandas: {}'.format(pd.__version__))  # Pandas version
        print('matplotlib: {}'.format(matplotlib.__version__))  # Matplotlib version
        print('seaborn: {}'.format(sns.__version__))  # seaborn version
        print('sklearn: {}'.format(sklearn.__version__))  # sklearn version
```

1.4.2 No Warnings

```
In [ ]: # No warning of any kind please!
        import warnings
        # will ignore any warnings
        warnings.filterwarnings("ignore")
```

1.4.3 Data is Here

```
In [ ]: # import data
```

1.5 ------

1.6 Data Exploration

1.6.1 Peak at Data

```
In [1]: # display data
```

1.6.2 Feature Statistics

Feature Describe

```
In [2]: # statistical summary
```

Feature Skew

```
In [3]: # Skewness of features
```

Class Distribution Let's take a look how each class is distributed..

```
In [4]: # Target Variable Class Distribution (Classification Problem)
```

1.6.3 Feature Visualization

Feature Spread

```
In [ ]: # boxplots of features
```

Feature Distribution

```
In [5]: # counts of categorical features, how many each feature occur in all observations
```

Feature Comparison

```
In [ ]: # comparing features with target variable
```

Feature Correlation

```
In [ ]: # correlation of Features
```

1.7 --------

1.8 Data Engineering

1.8.1 Observation Cleaning

In [6]:
```
# (Categoriacal Variables)

# Checking if any observation have more than 1 presence of _____ at same time or None

# Count for more than 1 presence
more_count = 0
# Count for none presence
none_count = 0
# total count
total = 0

#looping through each row of Soil Type area column
for index, row in ______.iterrows():
    # adding the values of each column of that row
    total = row.sum(axis=0)

    #checking greater than 1
    if total > 1:
        # if found, increment count by 1
        more_count =+ 1
        # reset the total
        total = 0
        # do not execute code below, start from top
        break

    #checking for none
    if total == 0:
        # if found, increment count by 1
        none_count =+ 1
        # reset the total
        total = 0

# priting results found
print('We have ', more_count, ' observations that shows presence in more than 1 _____.
print('We have ' ,none_count, ' observations that shows no presence in any _____.')
```

Handling Missing Values

In [ ]:
```
# will delete observation if it has any missing values in any of the features.
data.dropna()

# shape of the data after deleting missing entries
data.shape
```

Handling Duplicates

```
In [ ]: # deleting duplicates, except the first observation
        data.drop_duplicates(keep='first')

        # shape of the data after deleting duplicate entries
        data.shape
```

1.8.2 Dimentionality Reduction

Extra-Trees Classifier

```
In [ ]: # importing model for feature importance
        from sklearn.ensemble import ExtraTreesClassifier

        # passing the model
        model = ExtraTreesClassifier(random_state = 53)

        # feeding all our features to var 'X'
        X = data.iloc[:,:-1]
        # feeding our target variable to var 'y'
        y = data['']

        # training the model
        model.fit(X, y)

        # extracting feature importance from model and making a dataframe of it in descending 
        ETC_feature_importances = pd.DataFrame(model.feature_importances_, index = X.columns, 

        # removing traces of this model
        model = None

        # show top 10 features
        ETC_feature_importances.head(10)
```

Random Forest Classifier

```
In [ ]: # importing model for feature importance
        from sklearn.ensemble import RandomForestClassifier

        # passing the model
        model = RandomForestClassifier(random_state = 53)

        # training the model
        model.fit(X, y)

        # extracting feature importance from model and making a dataframe of it in descending 
```

```
RFC_feature_importances = pd.DataFrame(model.feature_importances_, index = X.columns, c

# removing traces of this model
model = None

# show top 10 features
RFC_feature_importances.head(10)
```

AdaBoost Classifier

```
In [ ]: # importing model for feature importance
        from sklearn.ensemble import AdaBoostClassifier

        # passing the model
        model = AdaBoostClassifier(random_state = 53)

        model.fit(X, y)

        # extracting feature importance from model and making a dataframe of it in descending o
        ADB_feature_importances = pd.DataFrame(model.feature_importances_, index = X.columns, c

        # removing traces of this model
        model = None

        ADB_feature_importances.head(10)
```

Gradient Boosting Classifier

```
In [ ]: # importing model for feature importance
        from sklearn.ensemble import GradientBoostingClassifier

        # passing the model
        model = GradientBoostingClassifier(random_state = 53)

        # training the model
        model.fit(X, y)

        # extracting feature importance from model and making a dataframe of it in descending o
        GBC_feature_importances = pd.DataFrame(model.feature_importances_, index = X.columns, c

        # removing traces of this model
        model = None

        # show top 10 features
        GBC_feature_importances.head(10)
```

1.8.3 Feature Scaling

1.8.4 Train-Test Split

1.9 -------

1.10 Model Evaluations

In [ ]:
```
### defining function for training models and measuring performance

# to measure performance
from sklearn.model_selection import cross_val_score

# for calculating time elapsed
import time

# fucntion
def model_evaluation(clf):

    # passing classifier to a variable
    clf = clf

    # records time
    t_start = time.time()
    # classifier learning the model
    clf = clf.fit(X_train, y_train)
    # records time
    t_end = time.time()


    # records time
    c_start = time.time()
    # Using 10 K-Fold CV on data, gives peroformance measures
    accuracy  = cross_val_score(clf, X_train, y_train, cv = 10, scoring = 'accuracy')
    f1_score = cross_val_score(clf, X_train, y_train, cv = 10, scoring = 'f1_macro')
    # records the time
    c_end = time.time()


    # calculating mean of all 10 observation's accuracy and f1, taking percent and rou
    acc_mean = np.round(accuracy.mean() * 100, 2)
    f1_mean = np.round(f1_score.mean() * 100, 2)


    # substracts end time with start to give actual time taken in seconds
    # divides by 60 to convert in minutes and rounds the answer to three decimal place
    # time in training
```

```
t_time = np.round((t_end - t_start) / 60, 3)
# time for evaluating scores
c_time = np.round((c_end - c_start) / 60, 3)


# Removing traces of classifier
clf = None


# returns performance measure and time of the classifier
print("The accuracy score of this classifier on our training set is", acc_mean,"%
      "minutes to evaluate cross validation and metric scores.")
```

1.10.1 Model 1

1.10.2 Model 2

1.10.3 Model 3

1.10.4 Model 4

1.10.5 Model 5

1.10.6 Model 6

1.10.7 Choosing Model

Out of 6 Models evaluated above and benchmark model, which performs better? Lets see all the scores of all the models in a table below:

Model	Accuracy	F1 Score	Train Time (m)	Evaluation Time (m)

1.11 ------

1.12 Testing Model

In [ ]:
```
# importing EM scores for model performance measure
from sklearn.metrics import accuracy_score, f1_score

# definning best chosen classifier
```

```
clf = ____

# training our model
clf = clf.fit(X_train, y_train)

# predicting unseen data
predict = clf.predict(X_test)

# calculating accuracy
accuracy = accuracy_score(y_test, predict)

# calculating f1 score
f1_score = f1_score(y_test, predict, average = 'macro')

# taking precentage and rounding to 3 places
accuracy = np.round(accuracy * 100, 3)
f1_score = np.round(f1_score * 100, 3)

# cleaning traces
clf = None

# results
print("The accuracy score of our final model ____ on our testing set is", accuracy,"% a
```

1.13 ------

1.14 Conclusion

1.15 ------

1.16 Notes

1.17 ------