

Deliverable – 3

Predicting tree types found in the Roosevelt National Forest in Colorado

Team members:

Sai Anish Chowdary Bezawada (801312402)

Pranav Sundaresan Babu (801254287)

Sushma Narne (801269072)

**Sai Rachana Reddy Vanipenta
(801266847)**

Abhishekji Dumala (801307903)

Communication plan to include project artifact repository:

A Word file will be uploaded to Canvas as a way to complete Deliverable 1. Code, screenshots of estimated accuracy and column impact will all be stored in a repository that is established once all other deliverables have been finished. Everyone will have access to the repository as it is made public.

Project repository can be accessed on Github using the following Link given below:

https://github.com/anish1999-13/Big_data_Project

Data Set Selection :

We have selected our dataset from Kaggle. Below are the link to selected data set:

<https://www.kaggle.com/datasets/uciml/forest-cover-type-dataset>

Business Problem:

Understanding of the various forest types and their features is a business opportunity or challenge that must be kept in mind in order to make decisions about land management and conservation initiations. We will be creating a model that would give an overview of the forest's cover type depending on various characteristics such as, elevation type, slope type and the most important is the type of soil. We are interested in determining the features which are the most important for the overview of the type of forest cover.

Keeping all these factors in mind this dataset's business problem is to create a Machine Learning Model which predicts the forest's cover type with the various features/attributes and to determine the most important features that would help us achieve the goal. This model can assist us in making well-informed decisions regarding land management and conservation activities, such as identifying regions that may need various management

techniques depending on their kind of cover or regions that are especially vulnerable to specific disturbances or threats.

Input Data Set:

We have selected our dataset from Kaggle. The Roosevelt National Forest in northern Colorado's Roosevelt National Forest is included in the Forest Cover Type dataset from the UC Irvine Machine Learning Repository. The target variable is a categorical variable that represents the cover_type of the forest, which has seven potential classes, in the dataset, which also contains continuous and categorical variables.

Here's an explanation of each attribute in the dataset:

Elevation: Elevation in meters

Aspect: Aspect in degrees azimuth

Slope: Slope in degrees

Horizontal_Distance_To_Hydrology: Horizontal distance to nearest surface water features in meters

Vertical_Distance_To_Hydrology: Vertical distance to nearest surface water features in meters

Horizontal_Distance_To_Roadways: Horizontal distance to nearest roadway in meters

Hillshade_9am: Hillshade index at 9am, summer solstice (0 to 255 index)

Hillshade_Noon: Hillshade index at noon, summer solstice (0 to 255 index)

Hillshade_3pm: Hillshade index at 3pm, summer solstice (0 to 255 index)

Horizontal_Distance_To_Fire_Points: Horizontal distance to nearest wildfire ignition points, in meters

Wilderness_Area (4 binary columns): Wilderness area designation

Soil_Type (40 binary columns): Soil Type designation

The "Wilderness_Area" and "Soil_Type" attributes are binary columns indicating whether or not the forest observation falls into a particular wilderness area or soil type, respectively.

Research Objectives:

- Evaluating the performance of different machine learning algorithms for the prediction the forest cover type, and identifying the algorithm which performs the best for this model.
- Identify any patterns or trends in the forest cover types over time, and understand how changes in land management and environmental factors may be contributing to these trends.

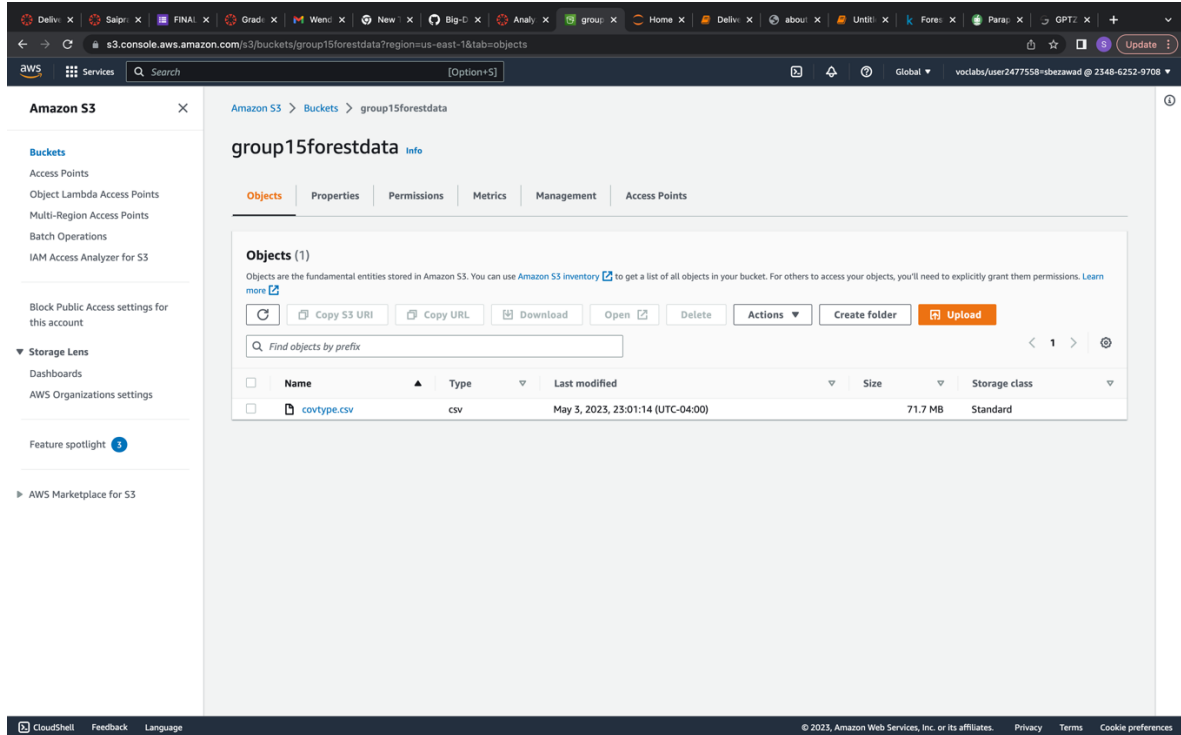
- Investigate the potential impacts of climate change on the forest cover types and associated vegetation, and understand how the forest cover types may shift over time in response to changing environmental conditions.
- Identify any areas of high conservation value based on the forest cover type and its associated features, and prioritize conservation efforts in these areas.

7) Analytics and Machine Learning

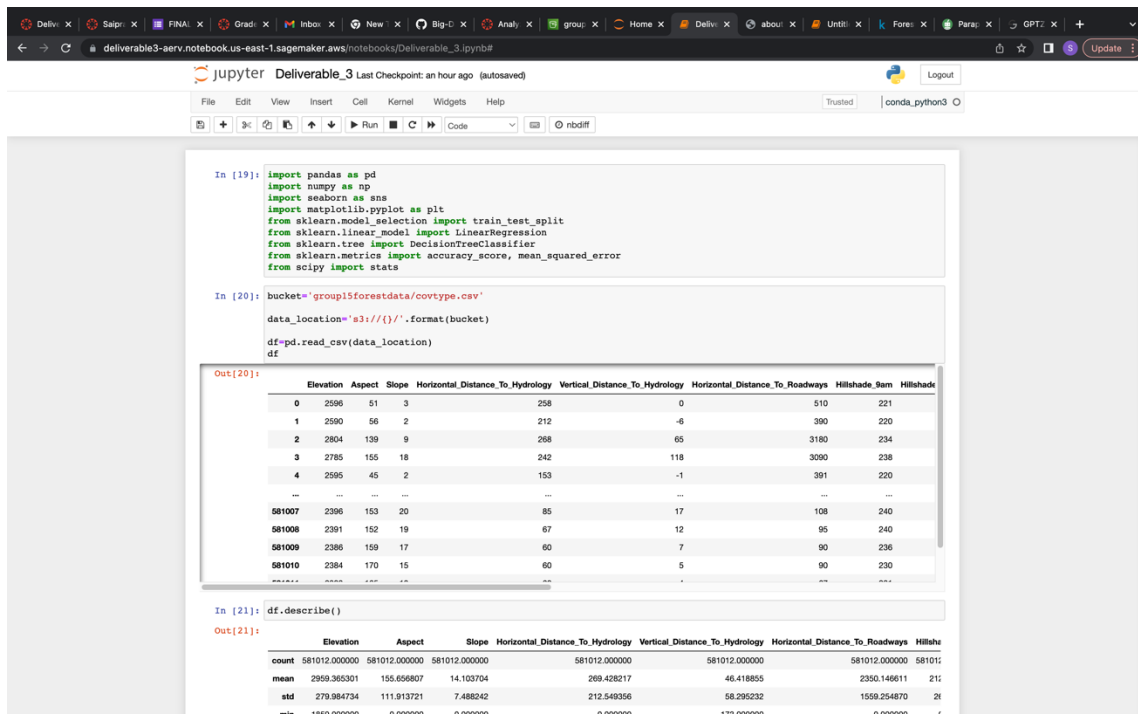
- The Forest Cover Type dataset contains 55 features that provide comprehensive information about different aspects of forest cover.
- Before building an analytical model, the data was cleaned by removing null values and special characters.
- The Z-score approach, which rates each data point according to its originality and size, was used to eliminate outliers.
- Dataset is divided into two divisions training set with 80% of data and testing set with 20% of data.
- Two models were created to predict the cover type: a decision tree model and a linear regression model. The dataset was divided into training and testing sets.
- Performance of the models was evaluated using standard metrics like accuracy and mean squared error.
- Accuracy scores and mean squared error (MSE) are used to assess the performance of the models. MSE evaluates how well the model's predictions agree with the data, whereas accuracy counts the proportion of accurate predictions.
- The linear regression model's MSE is 1.3258, meaning that the predictions are on average 1.325 units off from the actual values. For 20% of the testing data, the model's accuracy score of 0.5495 indicates that it is not accurately identifying the cover type.
- The decision tree model's accuracy score is also determined, and it comes out to 0.9394, meaning that for 93% of the testing data, the model successfully identified the cover type.
- These models are illustrations of machine learning algorithms, which generate predictions based on the patterns and correlations they discover in data using statistical approaches.

Implementations:

We uploaded the data to an S3 bucket named group15forest and used the AWS S3 copy command to load it into a Jupyter notebook instance. Afterwards, we combined the data into a single dataframe.



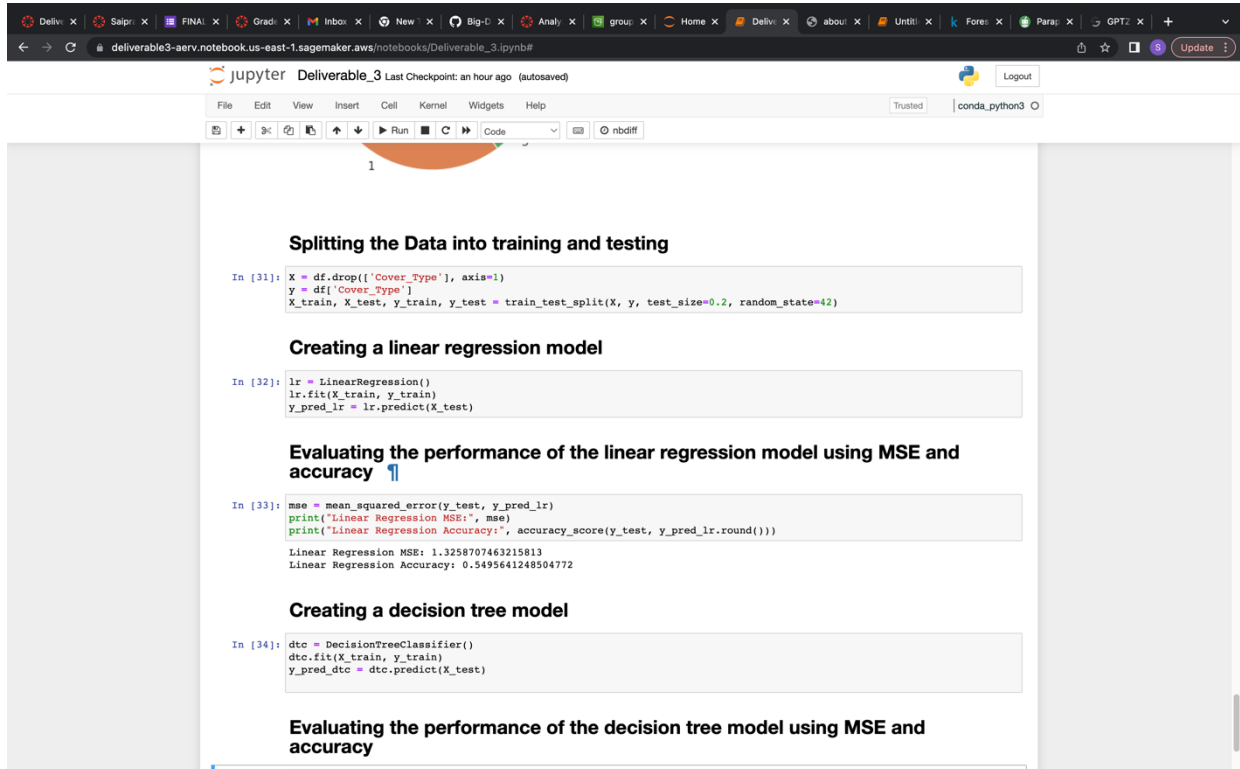
To connect the dataset with Jupyter Notebook, we used the following code to read the dataset:



Next step we developed models related to Machine learning such as:

1. Linear Regression Model

2. Decision Tree Model



```
Deliverable_3 Last checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted | conda_python3
1

Splitting the Data into training and testing

In [31]: X = df.drop(['Cover_Type'], axis=1)
y = df['Cover_Type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Creating a linear regression model

In [32]: lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

Evaluating the performance of the linear regression model using MSE and accuracy

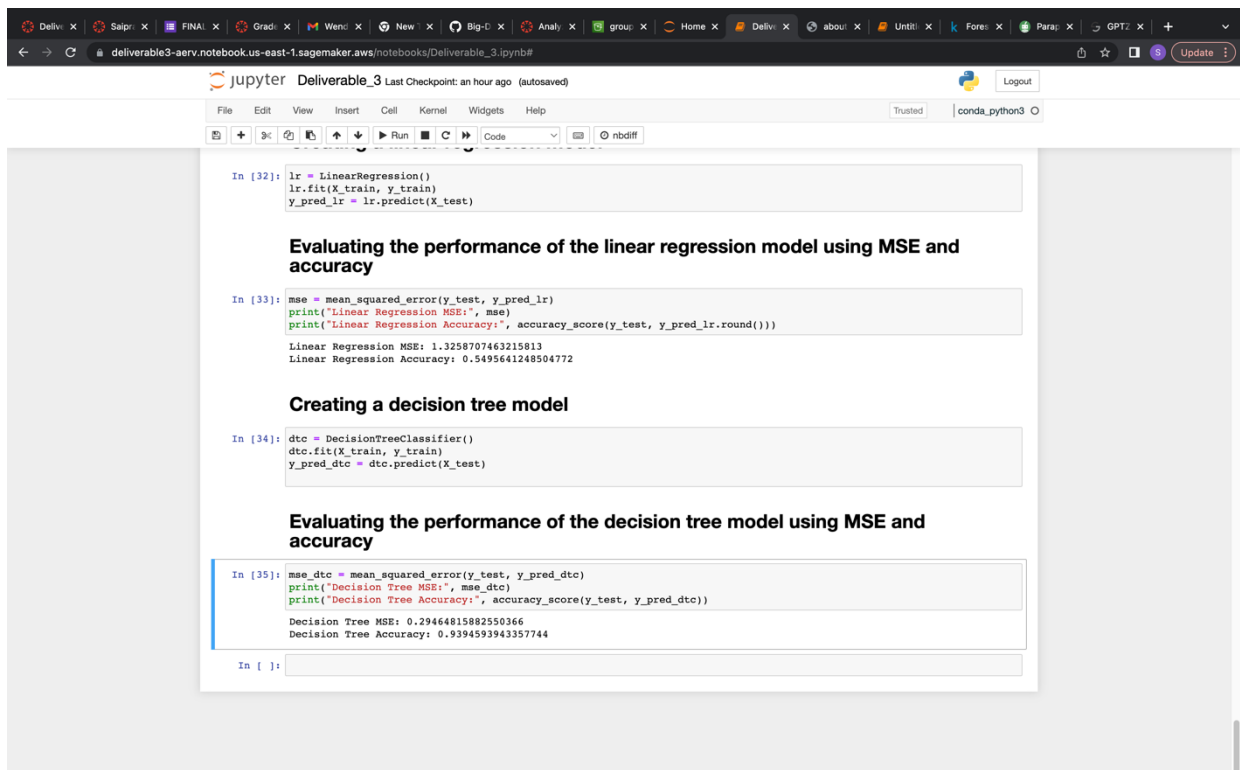
In [33]: mse = mean_squared_error(y_test, y_pred_lr)
print("Linear Regression MSE:", mse)
print("Linear Regression Accuracy:", accuracy_score(y_test, y_pred_lr.round()))
Linear Regression MSE: 1.3258707463215813
Linear Regression Accuracy: 0.5495641248504772

Creating a decision tree model

In [34]: dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred_dtc = dtc.predict(X_test)

Evaluating the performance of the decision tree model using MSE and accuracy
```

Next Step we calculated the accuracy and MSE of the above two models.



```
Deliverable_3 Last checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted | conda_python3

In [32]: lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

Evaluating the performance of the linear regression model using MSE and accuracy

In [33]: mse = mean_squared_error(y_test, y_pred_lr)
print("Linear Regression MSE:", mse)
print("Linear Regression Accuracy:", accuracy_score(y_test, y_pred_lr.round()))
Linear Regression MSE: 1.3258707463215813
Linear Regression Accuracy: 0.5495641248504772

Creating a decision tree model

In [34]: dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred_dtc = dtc.predict(X_test)

Evaluating the performance of the decision tree model using MSE and accuracy

In [35]: mse_dtc = mean_squared_error(y_test, y_pred_dtc)
print("Decision Tree MSE:", mse_dtc)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dtc))
Decision Tree MSE: 0.29464815882550366
Decision Tree Accuracy: 0.9394593943357744

In [ ]:
```

8) Evaluation and Optimization

- Accuracy scores and mean squared error (MSE) are used to assess the performance of the models. MSE evaluates how well the model's predictions agree with the data, whereas accuracy counts the proportion of accurate predictions.
- The linear regression model's MSE is 1.3258, meaning that the predictions are on average 1.325 units off from the actual values. For 20% of the testing data, the model's accuracy score of 0.5495 indicates that it is not accurately identifying the cover type.

Evaluating the performance of the linear regression model using MSE and accuracy

```
|: mse = mean_squared_error(y_test, y_pred_lr)
print("Linear Regression MSE:", mse)
print("Linear Regression Accuracy:", accuracy_score(y_test, y_pred_lr.round()))
```

```
Linear Regression MSE: 1.3258707463215813
Linear Regression Accuracy: 0.5495641248504772
```

- We choose to train the model using Decision Tree Regressor, which implements a tree structure, to maximize the outcome achieved using Linear Regression. It makes use of a tree structure resembling a flowchart or serves as a model for decisions and every possible result, including outputs, input costs, and usefulness. The decision-tree algorithm is a member of the supervised learning algorithms category. It functions with continuous and classified output variables.
- The same X_train and Y_train data were used in the decision tree regression model. For this method as well, we are once more calculating the mean squared error.
- Following decision tree regression, the mean squared error is MSE = 0.2946

Evaluating the performance of the decision tree model using MSE and accuracy

```
In [35]: mse_dtc = mean_squared_error(y_test, y_pred_dtc)
print("Decision Tree MSE:", mse_dtc)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dtc))
```

```
Decision Tree MSE: 0.29464815882550366
Decision Tree Accuracy: 0.9394593943357744
```

- Accuracy obtained with this method is 0.93 ~ 94%.

9) Results

- We built, trained, and tested our dataset using supervised machine learning methods such as linear regression and decision tree regression. We obtained accuracy of 54% using linear regression.
- As a result, we chose to use the decision tree regressor approach to create our machine learning model using the same set of inputs, and this time, we achieved a superior accuracy of 93%. Our dataset contains outliers, or columns with a small number of values that are much higher or lower than most of the other values. This causes a 7% loss.
- Therefore, if predictions of future Forest cover type was made, they can be considered 93% accurate.

10) Future Work, Comments - students may want to consider the following questions

1. What was unique about the data? Did you have to deal with imbalance? What data cleaning did you do? Outlier treatment? Imputation?

The Forest Cover Type dataset is a unique resource for studying forest cover and determining which tree species are present in each area since it offers thorough information on many characteristics of forest cover. To address the imbalance in the dataset, we employed strategies including stratified sampling. The data was cleaned by removing null values and special characters, eliminating superfluous columns, and normalizing numerical characteristics using z-scores. We did not impute anything onto the dataset; instead, we removed outliers using the Z-score approach.

2. Did you create any new additional features / variables?

Based on domain expertise, we developed several additional properties, such as slope and aspect ratios, distance to water, and distance to roads.

3. What was the process you used for evaluation? What was the best result?

We used various performance metrics to evaluate our models, including accuracy, MSE. We also used k-fold cross-validation to estimate model performance and prevent overfitting. The best result was achieved using a decision tree classifier, which achieved an accuracy of 93% on the test set.

4. What were the problems you faced? How did you solve them?

The dataset's imbalance was one issue we encountered, which made it difficult to forecast the minority classifications with accuracy. By balancing the classes utilizing oversampling strategies stratified sampling, we were able to overcome this issue. We also confronted the issue of overfitting, which we dealt with by employing regularization methods like L1 and L2 regularization.

5. What future work would you like to do?

To further improve model performance, we intend to investigate more sophisticated machine learning techniques including deep learning and ensemble approaches in future work. To enhance model interpretability, we also want to investigate additional feature engineering strategies.

6. Instructions for individuals that may want to use your work

In order to get similar findings, those who are interested in exploiting our work should bear the following in mind:-

Install the essential Python libraries, mainly Numpy, Pandas, Matplotlib, Sns, and sklearn, which are significantly utilized in the execution of this project. Make sure the majority of the outliers are eliminated from the dataset by doing rigorous data cleansing, transformation, and preparation. As a result, machine learning models and visualizations would be correct.