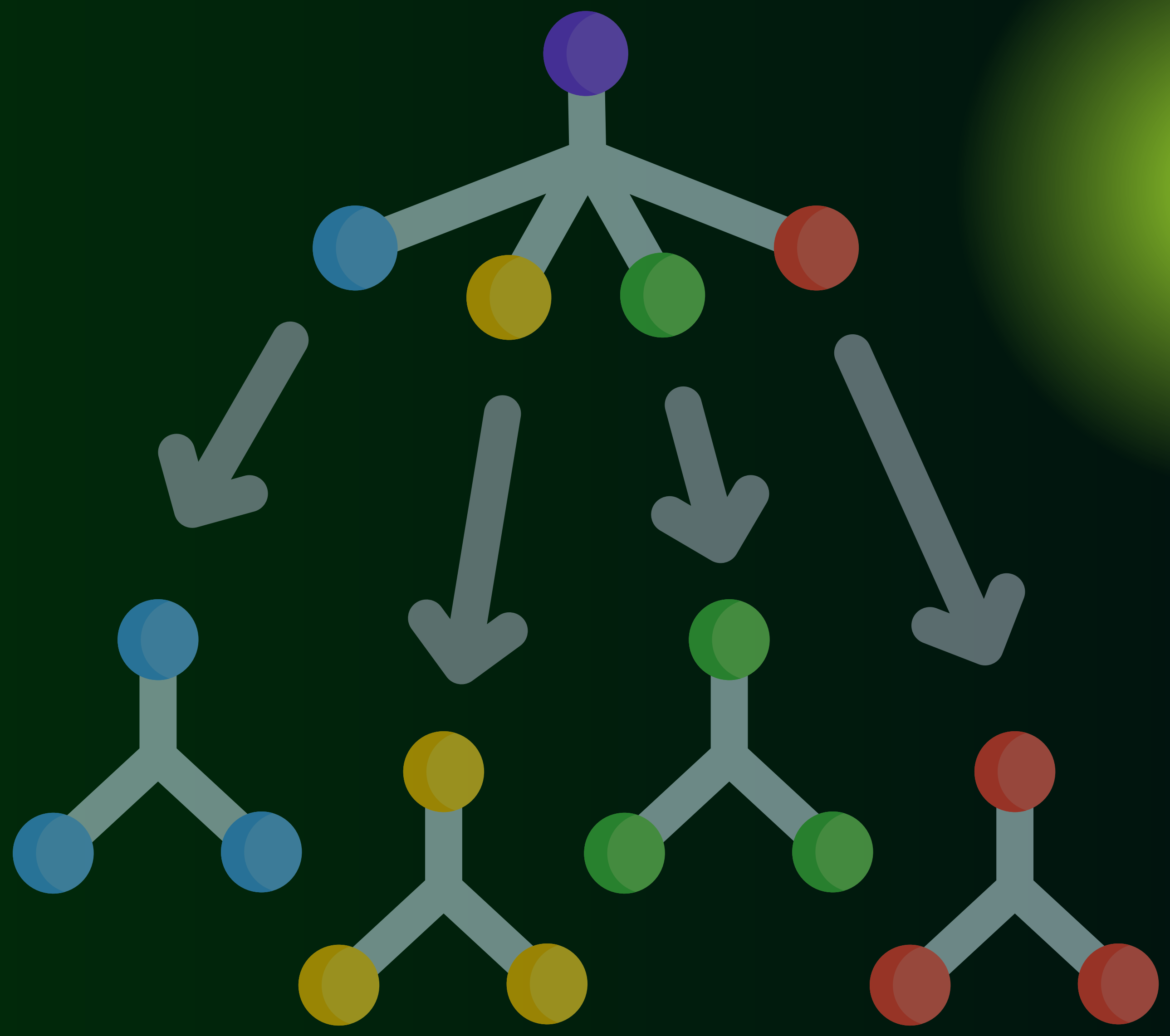
GROUP 7
RANDOM FOREST



PRESENTED BY:

SHREYA SARAF, 115
SUDHANSHU SHEKHAR, 119
JASHWANT NAYAK, 98
HIJAM JITESOR SINGH, 14
NITYA DOSAPATI, 91
HARSH CHAUDHARY, 12

Project Repository: *https://github.com/jit-hijam/randomforest_project_aml*

INTRODUCTION

01 What is Random Forest?

A Random Forest is an ensemble learning method where multiple decision trees are constructed and then they are merged to get a more accurate prediction

02 Applications

Classification and regression tasks

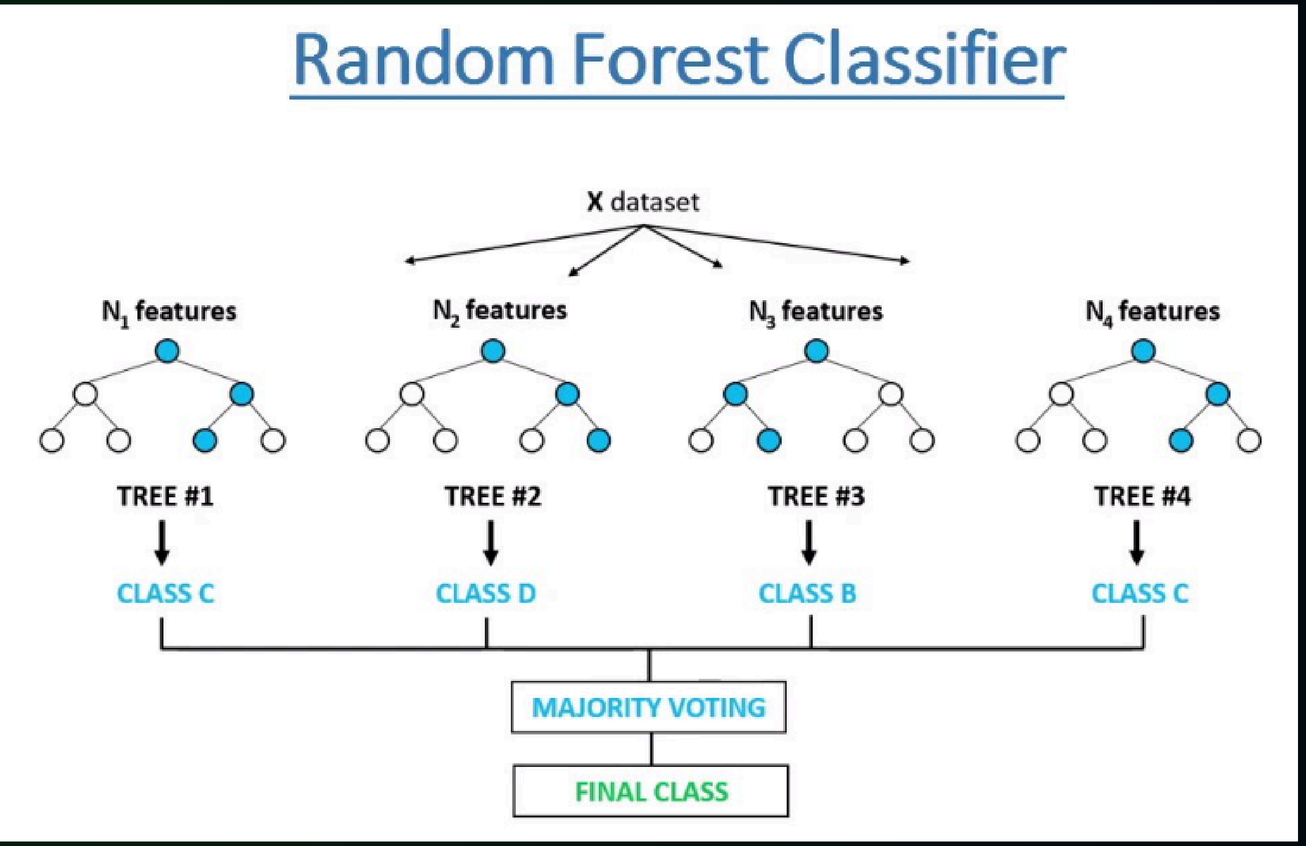
03 Why Random Forest?

Robustness and high accuracy

DETAILS OF THE ALGORITHM

Step1: Build the Forest:

- Pick Random features [RandomForestClassifier]
- Find the best split point among the k features to create a node (d). [rfc.fit(X_train, y_train)]
- Split the node into daughter nodes based on the best split.
- Repeat the process for n iterations to create n decision trees.



Step2: Making Predictions:

- Use test features and apply the rules of each tree to predict outcomes.
- Aggregate predictions by voting for each target.
- The target with the highest votes is selected as the final prediction.

AIM OF THE PROJECT

01 Predict car safety levels using the car evaluation dataset

02 Improve model accuracy by optimizing features and hyperparameters

03 Identify key influential features

DATASET OVERVIEW

```
# preview the dataset
df.head()
```

	0	1	2	3	4	5	6
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

Next steps: Generate code with df | View recommended plots | New interactive sheet

Rename column names

We can see that the dataset does not have proper column names. The columns are merely labelled as 0,1,2.... and so on. We should give proper names to the columns. I will do it as follows

```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   class     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Car Evaluation Dataset

01 Total records: 1,728 entries.

Grouping data points based on their similarities. Examples include k-means clustering and hierarchical clustering.

02 Rename Column Names

Features: buying, maint, doors, persons, lug_boot, safety, class

03 No Missing Values

There are 7 variables in the dataset. All the variables are of categorical data type. There are no missing values in the dataset.

DATA PRE-PROCESSING

Feature engineering: The process of transforming raw data into features that can be used to train machine learning models. Feature engineering involves selecting relevant features, creating new features, and transforming existing features

ENCODING CATEGORICAL VARIABLES USING ORDINAL ENCODING

SPLITTING DATASET INTO TRAINING (67%) AND TESTING (33%) SETS

SHAPE OF TRAINING/TESTING DATA: (1157, 6) AND (571, 6)

```
!pip install category_encoders
# import category encoders
import category_encoders as ce
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.4-py2.py3-none-any.whl.metadata (8.0 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encod
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoder
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->categor
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->categ
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0-
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pa
Downloading category_encoders-2.6.4-py2.py3-none-any.whl (82 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82.0/82.0 kB 4.3 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.4
```

```
# encode categorical variables with ordinal encoding

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'])

X_train = encoder.fit_transform(X_train)
```

s://colab.research.google.com/drive/12mLrqdQeAd81y37UkNz37XdC55BwAEl2#scrollTo=7rcW-DcyF-TS&printMode=true

/25, 8:19 PM .ipynb - Colab

```
X_test = encoder.transform(X_test)
```

```
X_train.head()
```

	buying	maint	doors	persons	lug_boot	safety
48	1	1	1	1	1	1
468	2	1	1	2	2	1
155	1	2	1	1	2	2
1721	3	3	2	1	2	2
1208	4	3	3	1	2	2

BUILDING RANDOM FOREST MODEL

Model 1: Default parameters (n_estimators=10), Accuracy: 92.64%.

```
Model accuracy score with 10 decision-trees : 0.9264
```

Model 2: Optimized parameters (n_estimators=200), Accuracy: 93.35%.

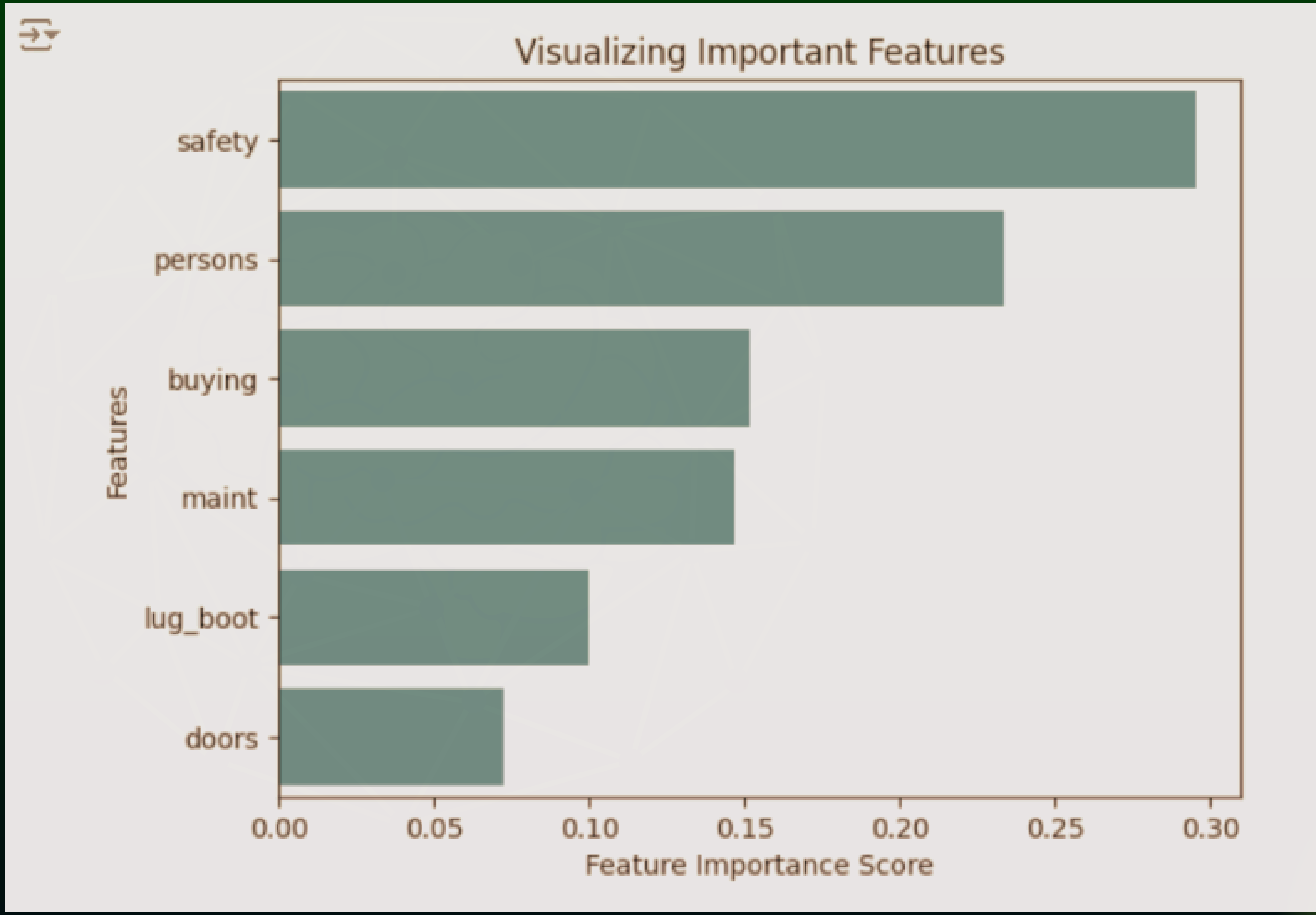
```
Model accuracy score with 200 decision-trees : 0.9335
```

The model accuracy score with 10 decision-trees is 0.9264 but the same with 200 decision-trees is 0.9335. So, as expected accuracy increases with number of decision-trees in the model.

IDENTIFYING KEY FEATURES

Top Features:
safety, persons, buying

Least Important:
doors



MODEL OPTIMIZATION

REMOVED DOORS, ACCURACY: 92.64%

REMOVING LUG_BOOT CAUSED SIGNIFICANT ACCURACY DROP

CONCLUSION: RETAIN MOST FEATURES FOR OPTIMAL PERFORMANCE

The second least important model is lug_boot. If we remove it from the model and rebuild the model, then the accuracy was found to be 0.8546. It is a significant drop in the accuracy. So, we will not drop it from the model.

Now, based on the above analysis we can conclude that our classification model accuracy is very good. Our model is doing a very good job in terms of predicting the class labels.

But, it does not give the underlying distribution of values. Also, it does not tell anything about the type of errors our classifier is making.

Now, we will drop the least important feature doors from the model, rebuild the model and check its effect on accuracy

```
# declare feature vector and target variable

X = df.drop(['class', 'doors'], axis=1)
y = df['class']

# split data into training and testing sets

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

Now, we will build the random forest model and check accuracy

```
# encode categorical variables with ordinal encoding

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'persons', 'lug_boot', 'safety'])

X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)

# instantiate the classifier with n_estimators = 200

clf = RandomForestClassifier(random_state=0)

# fit the model to the training set

clf.fit(X_train, y_train)

# Predict on the test set results

y_pred = clf.predict(X_test)

# Check accuracy score

print('Model accuracy score with doors variable removed : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

Model accuracy score with doors variable removed : 0.9264
```

CONFUSION MATRIX AND CLASSIFICATION REPORT

```
Confusion matrix

 [[104  12  10   3]
 [  0  18   0   2]
 [ 10   0 387   0]
 [  3   2   0  20]]
```

	precision	recall	f1-score	support
acc	0.89	0.81	0.85	129
good	0.56	0.90	0.69	20
unacc	0.97	0.97	0.97	397
vgood	0.80	0.80	0.80	25
accuracy			0.93	571
macro avg	0.81	0.87	0.83	571
weighted avg	0.93	0.93	0.93	571

Overall model accuracy: 93%

KEY TAKEAWAYS

01 Random Forest achieved high accuracy (93.35%)

02 safety is the most significant feature

03 Feature selection and hyperparameter tuning are critical for performance improvement

PRO AND CONS

Aspect	Pros	Cons
Accuracy	High accuracy due to ensemble learning	Computationally expensive for large datasets
Robustness	Handles noise and missing values effectively	High memory usage
Interpretation	Provides feature importance for insights	Less interpretable compared to single decision trees
Versatility	Works for both classification and regression tasks	Potential overfitting with excessive trees

UTILITY IN BUSINESS ANALYTICS

Area	Usage
Prediction	Forecasting sales, demand, or customer churn
Classification	Identifying customer segments or fraud detection
Feature Selection	Determining important factors impacting business outcomes
Handling Complex Data	Works well with large, high-dimensional datasets

UTILITY IN BUSINESS SCENARIOS

Scenario	Example
Customer Segmentation	Classifying customers into high-value, medium-value, and low-value **(Amazon)**
Fraud Detection	Identifying fraudulent transactions in banking **(PayPal)**
Predictive Maintenance	Forecasting machinery breakdowns in manufacturing **(General Electric)**
Supply Chain Optimization	Predicting demand for inventory management **(Walmart)**

COMPARISON WITH SIMILAR ALGORITHMS

Algorithm	Advantages Over Random Forest	Disadvantages Compared to Random Forest
Decision Tree	Easier to interpret, faster for small datasets	Prone to overfitting, less accurate
Gradient Boosting	Higher accuracy in some cases, better optimization	Slower training, sensitive to hyperparameters
K-Nearest Neighbors	Simple and intuitive, no training phase	Inefficient for large datasets, no feature importance
Support Vector Machine	Works well with smaller datasets and linear data	Difficult to scale to large datasets

REFERENCES (BLOGS AND VIDEO LINKS)

Random Forest Algorithm in Machine Learning
https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

What is random forest?
https://www.ibm.com/think/topics/random-forest

Random Forest Classifier using Scikit-learn
https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

Video tutorials in YouTube:
https://youtu.be/eM4uJ6XGnSM?si=erpSmMNeRApsK17W
https://youtu.be/gkXX4h3qYm4?si=ysaAXlgeruMzCr_S

ANY QUESTIONS ?