**Credit Risk Prediction using Decision Trees and Random Forest**

**Introduction**

Credit risk prediction is one of the most important tasks in financial institutions. Identifying whether a transaction is legitimate or fraudulent helps prevent financial losses and maintain customer trust. In this project, we use Decision Trees and Random Forests to predict fraudulent transactions. We also perform hyperparameter tuning using GridSearchCV to improve model performance.

**Step 1: Importing Dataset**

We begin by loading the dataset using Pandas:

df = pd.read\_csv("creditcard.csv")

* Checked for missing values:
* df.isnull().sum()
* Inspected first few rows:
* print(df.head())
* Data types:
* df.dtypes
* Class distribution:
* df['Class'].value\_counts()

**Step 2: Balancing the Dataset**

The dataset is highly imbalanced, with legitimate transactions far outnumbering fraudulent ones.  
To balance:

legit = df[df.Class == 0]

fraud = df[df.Class == 1]

legit\_sample = legit.sample(n=len(fraud))

df2 = pd.concat([legit\_sample, fraud], axis=0)

This ensures an equal number of legitimate and fraudulent cases.

**Step 3: Splitting Data into Features and Target**

* Features (X): All columns except "Class".
* Target (y): "Class".

x = df2.drop("Class", axis='columns')

y = df2['Class']

We split into training and testing sets (80/20 ratio):

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, test\_size=0.2, stratify=y, random\_state=2)

**Step 4: Base Models**

**Decision Tree Classifier**

dt = DecisionTreeClassifier(random\_state=42)

dt.fit(x\_train, y\_train)

y\_pred\_dt = dt.predict(x\_test)

* Accuracy & Classification Report:
* accuracy\_score(y\_test, y\_pred\_dt)
* classification\_report(y\_test, y\_pred\_dt)

**Random Forest Classifier**

rf = RandomForestClassifier(random\_state=42)

rf.fit(x\_train, y\_train)

y\_pred\_rf = rf.predict(x\_test)

* Accuracy & Classification Report:
* accuracy\_score(y\_test, y\_pred\_rf)
* classification\_report(y\_test, y\_pred\_rf)

**Step 5: Hyperparameter Tuning**

We improve model performance using **GridSearchCV** with 5-fold cross-validation.

**Decision Tree Tuning**

Parameters:

dt\_params = {

'max\_depth': [3, 5, 10, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'criterion': ['gini', 'entropy']}

Best parameters:

{'criterion': 'gini', 'max\_depth': 3,

'min\_samples\_leaf': 2,

'min\_samples\_split': 2}

**Random Forest Tuning**

Parameters:

rf\_params = {

'n\_estimators': [50, 100, 200],

'max\_depth': [3, 5, 10, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

Best parameters:

{'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10, 'n\_estimators': 200}

Best Cross-Validation Accuracy: 94.66%

**Step 6: Feature Importance**

Both Decision Trees and Random Forests provide **feature importance scores**, which help identify which variables contribute most to distinguishing between legitimate and fraudulent transactions.

**Decision Tree (Tuned)**

* Feature importance can be accessed using:

best\_dt.feature\_importances\_

**Key Insights:**

* Highly dependent on **V15**.
* Minor contributions from **V4, V9, V11, V21, V28**.
* Most other features have negligible impact (<1%).

**Random Forest (Tuned)**

* Feature importance can be accessed using:

best\_rf.feature\_importances\_

**Key Insights:**

* Major contributors include **V15, V11, V5, V17, and V13**.
* Moderate contributions from **V3, V12, V18, V10**.
* Minimal impact from features like **Time, Amount, V24–V28**.

**Interpretation:**

* Decision Tree relies heavily on a single feature, making it less robust.
* Random Forest leverages multiple features, improving generalization and accuracy.

**Conclusion**

* **Decision Tree (Base):** Accuracy ≈ **86%**
* **Random Forest (Base):** Accuracy ≈ **92%**
* **Decision Tree (Tuned):** Accuracy improved after hyperparameter optimization (**93.5% CV**).
* **Random Forest (Tuned):** Outperformed Decision Tree with (**94.6% CV accuracy**).