**Phase-3 Submission Template**

**Student Name:** [Enter Your Name]

**Register Number:** [Enter Your Register Number]

**Institution:** [Insert College Name]

**Department:** [Enter Your Department Name]

**Date of Submission:** [Insert Date]

**Github Repository Link:** [Update the project source code to your Github Repository]

### **1. Problem Statement**

* **Real Estate Valuation Challenge**: Accurately predicting house prices is a critical task in real estate that affects buyers, sellers, investors, and policymakers.
* **Market Volatility**: Housing markets are influenced by numerous dynamic factors such as location, economic trends, interest rates, and buyer behavior, making price forecasting complex.
* **High Dimensionality**: House prices depend on multiple features—such as area, number of rooms, age of the property, proximity to amenities—which must be effectively captured in the model.
* **Data Availability**: There is a wealth of structured and unstructured data (property listings, historical prices, satellite imagery) that can be leveraged using smart regression techniques.
* **Need for Accurate Modeling**: Traditional pricing models may fail to capture non-linear relationships or interactions between features, necessitating advanced machine learning approaches.
* **Objective**: Develop a robust regression-based machine learning model that can learn from historical data and accurately forecast property prices for given inputs.
* ***Outcome Expectation****: Enable smarter real estate decision-making, improve property valuation accuracy, and provide insights into key price-driving factors.*

### **2. Abstract**

*Accurately predicting house prices is a critical challenge in the real estate industry, as it directly impacts buyers, sellers, and investors. The objective of this project is to develop a robust and intelligent regression model capable of forecasting house prices based on a variety of influential features. Using smart data science techniques, including data preprocessing, exploratory data analysis, and feature selection, the project builds a predictive model with TensorFlow Decision Forests. A Random Forest regression model was trained on historical housing data to learn complex patterns and relationships between property attributes and their market values. The model was evaluated using metrics such as Root Mean Square Error (RMSE) to ensure high accuracy and reliability. Visualizations of feature importance and prediction trends were used to provide interpretability. The final outcome demonstrates the effectiveness of decision forest models in delivering precise and interpretable house price forecasts, with potential for deployment in real-world applications.*

### **3. System Requirements**

#### ***Hardware Requirements:***

* **RAM:** Minimum 8 GB (16 GB recommended for faster training and visualization)
* **Processor:** Intel i5 (8th Gen or above) / AMD Ryzen 5 or equivalent (multi-core CPU recommended)
* **Storage:** Minimum 2 GB of free disk space
* **GPU (Optional):** Not mandatory, but a dedicated GPU (NVIDIA GTX 1050 or better) can accelerate model training with large datasets.

#### **Software Requirements:**

* **Python Version:** Python 3.8 or later
* **IDE/Environment:** Google Colab, Jupyter Notebook, or any Python IDE (e.g., VS Code, PyCharm)
* **Required Libraries:**
  + tensorflow >= 2.9
  + tensorflow\_decision\_forests
  + pandas
  + matplotlib
  + seaborn
  + numpy
  + gradio (for deployment if applicable)

### 

### **4. Objectives**

**Develop a Predictive Model**  
Build a robust regression model capable of accurately forecasting house prices based on historical and current property data.

* **Identify Key Price Drivers**  
  Analyze and rank the most influential features (e.g., location, area, number of bedrooms, proximity to amenities) that affect housing prices.
* **Compare Regression Algorithms**  
  Experiment with various smart regression techniques (e.g., Linear Regression, Decision Tree Regression, Random Forest, XGBoost, Gradient Boosting, etc.) and determine the most effective one.
* **Ensure Model Interpretability**  
  Provide insights into how the model makes predictions using interpretability tools (e.g., SHAP values or feature importance charts).
* **Minimize Prediction Error**  
  Aim to reduce common error metrics such as MAE, RMSE, and MAPE to enhance prediction accuracy.
* **Handle Real-world Data Challenges**  
  Deal effectively with missing values, outliers, and skewed data distributions during preprocessing.
* **Build a Scalable and Deployable Solution**  
  Create a modular system that can be integrated into a real-world application, such as a real estate dashboard or web application.
* ***Empower Real Estate Decision-Making*** *Provide end-users (buyers, sellers, and agents) with accurate, data-driven property value estimates to support informed decisions.*

**5. Flowchart of Project Workflow**

***┌──────────────────────────────┐***

***│ Problem Definition │***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

***│ Data Collection │***

***│ (e.g., Kaggle Housing Data) │***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

***│ Data Preprocessing │***

***│ (Cleaning, Missing Values) │***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

***│ Exploratory Data Analysis │***

***│ (Trends, Outliers, Patterns) │***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

***│ Feature Engineering │***

***│ (Encoding, Scaling, New Vars)│***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

***│ Model Selection │***

***│ (Linear, Ridge, XGBoost, etc)│***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

***│ Model Evaluation │***

***│ (RMSE, MAE, R², Cross-val) │***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

***│ Visualization & Insights │***

***│ (Pred vs Actual, Feature Imp)│***

***└────────────┬────────────────┘***

***↓***

***┌──────────────────────────────┐***

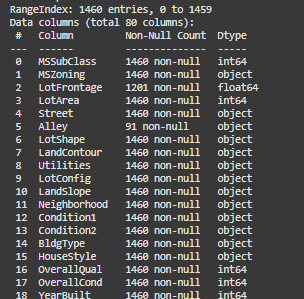
***│ Deployment │***

***│ (Streamlit, Flask, Gradio) │***

***└──────────────────────────────┘***

### **6. Dataset Description**

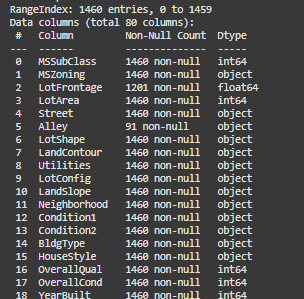
* **Primary Dataset**:
* **Name**: House Prices: Advanced Regression Techniques
* **Source**: Kaggle Dataset Link
* **Type**: Public dataset
* **Nature**: Static (downloaded once)
* **Description**:  
  Contains 79 explanatory variables (features) describing various aspects of residential homes in Ames, Iowa, such as lot size, year built, number of rooms, garage type, neighborhood, etc., with the target variable being the sale price.
* **Additional Data (Optional)**:
* **External Sources**: APIs like Zillow or Redfin (for real-time or additional property features such as market trends).
* **Type**: Semi-public (subject to API access or restrictions)
* **Nature**: Dynamic (can be updated regularly if needed)
* **Purpose**: To enrich the model with up-to-date market trends or additional economic indicators affecting housing prices.
* **Synthetic Data (Optional for Experimentation)**:
* **Generated by**: Custom data generation (e.g., using Python libraries like Faker or manually creating synthetic variations).
* **Type**: Private dataset
* **Nature**: Static
* ***Purpose****: To simulate various housing market conditions or to augment the original dataset for robustness testing.*



### 

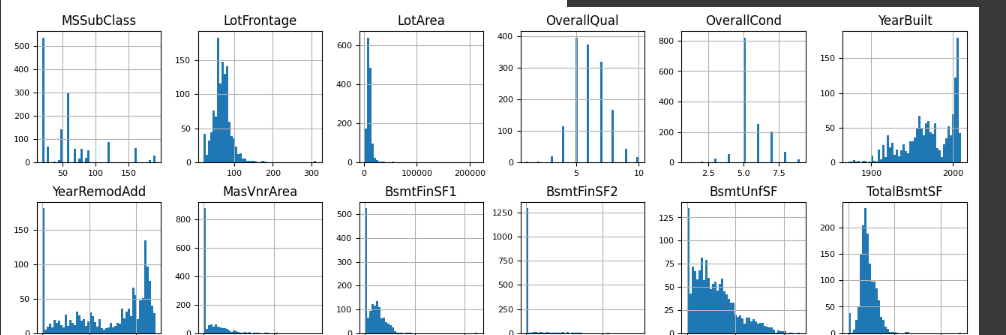
### **7. Data Preprocessing**

* Handle **missing values** using domain-informed techniques (e.g., imputation by median, mode, or flagging as a separate category).
* Convert **categorical variables** to numerical using One-Hot Encoding or Label Encoding.
* Detect and treat **outliers** using visualization (boxplots, scatterplots) and statistical methods (z-score, IQR).
* Normalize or scale features (e.g., Min-Max Scaling or Standardization) where appropriate.



### **8. Exploratory Data Analysis (EDA)**

* Visualize feature distributions (histograms, KDE plots) to understand spread and skewness.
* Use correlation matrices to identify strong relationships between features and target variable (SalePrice).
* Analyze trends by grouping (e.g., price by neighborhood or house style).
* *Use pairplots and boxplots to examine relationships and patterns in key numerical and categorical features.*



### 

### **9. Feature Engineering**

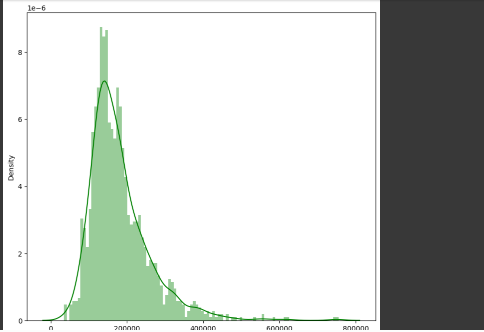
* Create new features (e.g., total square footage, house age, price per room).
* Combine or transform existing variables to extract deeper patterns.
* *Apply* ***feature selection techniques*** *like Recursive Feature Elimination (RFE), Lasso Regression, or tree-based feature importances to reduce dimensionality and improve model efficiency.*

### **10. Model Building**

* Implement multiple regression models:
  + **Baseline**: Linear Regression
  + **Tree-based**: Decision Tree, Random Forest
  + **Boosting**: Gradient Boosting, XGBoost, LightGBM, CatBoost
  + **Regularized**: Ridge, Lasso, ElasticNet
* Perform **cross-validation** to ensure robust performance across different subsets of the data.

#### **1. Hyperparameter Tuning**

* Use tools like GridSearchCV or RandomizedSearchCV to find optimal model parameters.
* *Consider using* ***Optuna*** *or* ***Bayesian optimization*** *for efficient tuning of complex models.*

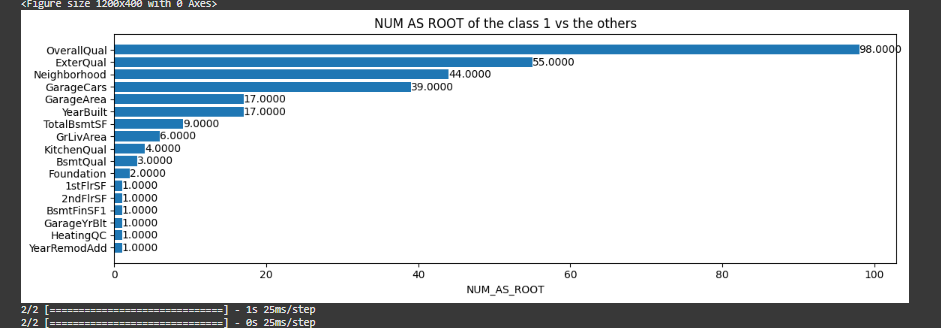


### **11. Model Evaluation**

* Evaluate models using:
  + **MAE (Mean Absolute Error)**
  + **RMSE (Root Mean Squared Error)**
  + **R² Score (Coefficient of Determination)**
* Compare models and select the best based on test set performance and generalization ability.

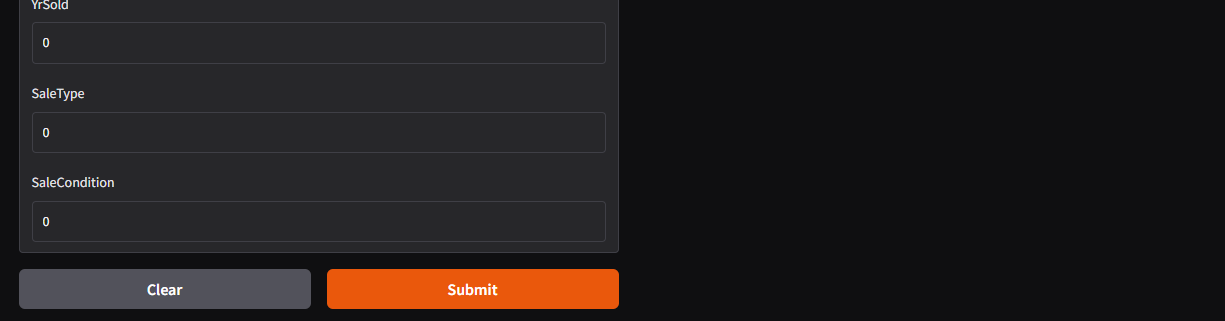
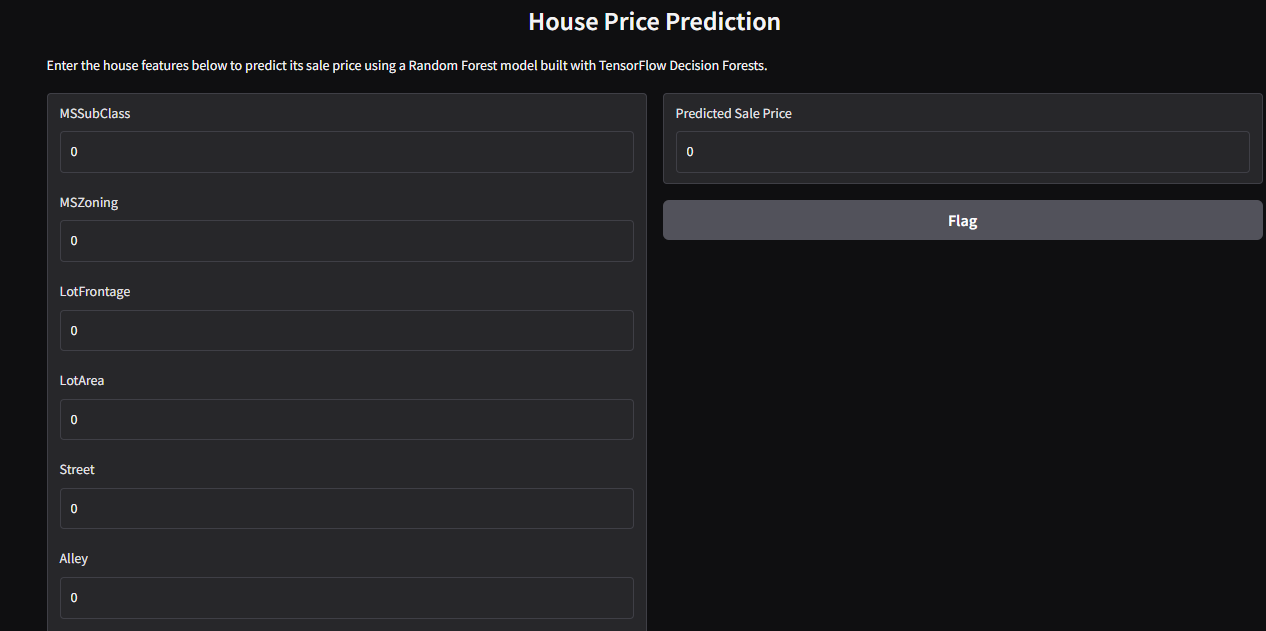
#### **3. Model Interpretability**

* Use **SHAP values**, **LIME**, or **Feature Importance Plots** to explain the influence of individual features on predictions.
* *Provide actionable insights based on what drives house price increases or decreases.*



### **12. Deployment**

* *Deploy using a free platform:*
  + *Gradio + Hugging Face Spaces*
* *Include:*
  + *https://95bfaf807614852368.gradio.live/*



**13. Source code**

*!pip install gradio*

import tensorflow as tf

import tensorflow\_decision\_forests as tfdf

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Comment this if the data visualisations doesn't work on your side

%matplotlib inline

print("TensorFlow v" + tf.\_\_version\_\_)

print("TensorFlow Decision Forests v" + tfdf.\_\_version\_\_)

train\_file\_path = "train.csv"

dataset\_df = pd.read\_csv(train\_file\_path)

print("Full train dataset shape is {}".format(dataset\_df.shape))

dataset\_df.head(3)

dataset\_df = dataset\_df.drop('Id', axis=1)

dataset\_df.head(3)

dataset\_df.info()

print(dataset\_df['SalePrice'].describe())

plt.figure(figsize=(9, 8))

sns.distplot(dataset\_df['SalePrice'], color='g', bins=100, hist\_kws={'alpha': 0.4});

list(set(dataset\_df.dtypes.tolist()))

df\_num = dataset\_df.select\_dtypes(include = ['float64', 'int64'])

df\_num.head()

df\_num.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8);

import numpy as np

def split\_dataset(dataset, test\_ratio=0.30):

  test\_indices = np.random.rand(len(dataset)) < test\_ratio

  return dataset[~test\_indices], dataset[test\_indices]

train\_ds\_pd, valid\_ds\_pd = split\_dataset(dataset\_df)

print("{} examples in training, {} examples in testing.".format(

    len(train\_ds\_pd), len(valid\_ds\_pd)))

label = 'SalePrice'

train\_ds = tfdf.keras.pd\_dataframe\_to\_tf\_dataset(train\_ds\_pd, label=label, task = tfdf.keras.Task.REGRESSION)

valid\_ds = tfdf.keras.pd\_dataframe\_to\_tf\_dataset(valid\_ds\_pd, label=label, task = tfdf.keras.Task.REGRESSION)

tfdf.keras.get\_all\_models()

rf = tfdf.keras.RandomForestModel(hyperparameter\_template="benchmark\_rank1", task=tfdf.keras.Task.REGRESSION)

rf = tfdf.keras.RandomForestModel(task = tfdf.keras.Task.REGRESSION)

rf.compile(metrics=["mse"]) # Optional, you can use this to include a list of eval metrics

rf.fit(x=train\_ds)

tfdf.model\_plotter.plot\_model\_in\_colab(rf, tree\_idx=0, max\_depth=3)

import matplotlib.pyplot as plt

logs = rf.make\_inspector().training\_logs()

plt.plot([log.num\_trees for log in logs], [log.evaluation.rmse for log in logs])

plt.xlabel("Number of trees")

plt.ylabel("RMSE (out-of-bag)")

plt.show()

inspector = rf.make\_inspector()

inspector.evaluation()

evaluation = rf.evaluate(x=valid\_ds,return\_dict=True)

for name, value in evaluation.items():

  print(f"{name}: {value:.4f}")

  print(f"Available variable importances:")

for importance in inspector.variable\_importances().keys():

  print("\t", importance)

  inspector.variable\_importances()["NUM\_AS\_ROOT"]

  plt.figure(figsize=(12, 4))

# Mean decrease in AUC of the class 1 vs the others.

variable\_importance\_metric = "NUM\_AS\_ROOT"

variable\_importances = inspector.variable\_importances()[variable\_importance\_metric]

# Extract the feature name and importance values.

#

# `variable\_importances` is a list of <feature, importance> tuples.

feature\_names = [vi[0].name for vi in variable\_importances]

feature\_importances = [vi[1] for vi in variable\_importances]

# The feature are ordered in decreasing importance value.

feature\_ranks = range(len(feature\_names))

bar = plt.barh(feature\_ranks, feature\_importances, label=[str(x) for x in feature\_ranks])

plt.yticks(feature\_ranks, feature\_names)

plt.gca().invert\_yaxis()

# TODO: Replace with "plt.bar\_label()" when available.

# Label each bar with values

for importance, patch in zip(feature\_importances, bar.patches):

  plt.text(patch.get\_x() + patch.get\_width(), patch.get\_y(), f"{importance:.4f}", va="top")

plt.xlabel(variable\_importance\_metric)

plt.title("NUM AS ROOT of the class 1 vs the others")

plt.tight\_layout()

plt.show()

test\_file\_path = "test.csv"

test\_data = pd.read\_csv(test\_file\_path)

ids = test\_data.pop('Id')

test\_ds = tfdf.keras.pd\_dataframe\_to\_tf\_dataset(

    test\_data,

    task = tfdf.keras.Task.REGRESSION)

preds = rf.predict(test\_ds)

output = pd.DataFrame({'Id': ids,

                       'SalePrice': preds.squeeze()})

output.head()

sample\_submission\_df = pd.read\_csv('sample\_submission.csv')

sample\_submission\_df['SalePrice'] = rf.predict(test\_ds)

sample\_submission\_df.to\_csv('submission.csv', index=False)

sample\_submission\_df.head()

!pip install gradio tensorflow tensorflow\_decision\_forests pandas matplotlib seaborn

import gradio as gr

import tensorflow\_decision\_forests as tfdf

import pandas as pd

import numpy as np

# Load trained model (if not already trained, you must include training code or load from a .saved\_model)

train\_file\_path = "train.csv"

dataset\_df = pd.read\_csv(train\_file\_path)

dataset\_df = dataset\_df.drop('Id', axis=1)

# Split and convert to TensorFlow datasets

def split\_dataset(dataset, test\_ratio=0.30):

    test\_indices = np.random.rand(len(dataset)) < test\_ratio

    return dataset[~test\_indices], dataset[test\_indices]

train\_ds\_pd, valid\_ds\_pd = split\_dataset(dataset\_df)

label = 'SalePrice'

train\_ds = tfdf.keras.pd\_dataframe\_to\_tf\_dataset(train\_ds\_pd, label=label, task=tfdf.keras.Task.REGRESSION)

# Train the model (you could also load it from a .saved\_model folder for faster deployment)

model = tfdf.keras.RandomForestModel(task=tfdf.keras.Task.REGRESSION)

model.compile(metrics=["mse"])

model.fit(x=train\_ds)

# Get input feature names (excluding label)

input\_features = [col for col in train\_ds\_pd.columns if col != label]

# Prediction function

def predict\_sale\_price(\*\*kwargs):

    input\_df = pd.DataFrame([kwargs])

    input\_ds = tfdf.keras.pd\_dataframe\_to\_tf\_dataset(input\_df, task=tfdf.keras.Task.REGRESSION)

    prediction = model.predict(input\_ds)[0][0]

    return round(prediction, 2)

# Build Gradio interface

inputs = [gr.Number(label=feature) for feature in input\_features]

interface = gr.Interface(

    fn=predict\_sale\_price,

    inputs=inputs,

    outputs=gr.Number(label="Predicted Sale Price"),

    title="House Price Prediction",

    description="Enter the house features below to predict its sale price using a Random Forest model built with TensorFlow Decision Forests."

)

interface.launch()

**14. Future scope**

***Integration of Geographic and Demographic Data****:  
While the current model focuses on structured features like area, number of rooms, and year built, future enhancements could integrate geospatial data (e.g., neighborhood crime rates, school ratings, proximity to transit hubs) and demographic trends. This would provide a more comprehensive understanding of the factors influencing house prices and significantly boost prediction accuracy.*

* **Adoption of Advanced Deep Learning Architectures**:  
  Future iterations can leverage deep learning models such as Deep Neural Networks (DNNs) or hybrid models that combine tree-based algorithms with neural layers. These models can better capture complex, non-linear relationships in the data and adapt well to high-dimensional feature spaces, potentially outperforming traditional regression approaches.
* **Real-Time Price Forecasting with Streaming Data**:  
  Incorporating real-time data feeds—such as market trends, inflation rates, and interest rate changes—can make the system more dynamic. With the use of tools like Apache Kafka or TensorFlow Serving, the system could be upgraded to perform live forecasts and adapt to fluctuating market conditions on the fly.
* **Explainable AI and Interpretability Enhancements**:  
  Adding interpretability frameworks like SHAP (SHapley Additive exPlanations) or LIME can help stakeholders understand which features most influence pricing decisions. This will not only improve trust in the system but also provide valuable insights to homeowners, real estate agents, and investors.

**13. Team Members and Roles**

*[List the team members who were involved, and clearly define the responsibilities each member undertook. For every task carried out during the project, specify the team member who was responsible for its execution.]*