Slide 1: Title Slide

- **Coding**: No specific code for this introductory slide, but you might mention that Python is the primary programming language used for the project.

Slide 2: Introduction

- **Coding**: Explain that the dataset was loaded and inspected as the first step to understand its structure and key variables.

```
import pandas as pd
df = pd.read_csv('housing.csv')  # Load the dataset
print(df.head())              # Display first few rows to inspect data
```

Slide 3: Problem Statement

- **Coding**: Here, you can explain that the problem was defined by analyzing the `median_house_value` column as the target variable and identifying factors that may influence it, such as `median_income` and `housing_median_age`.

Slide 4: Objectives

- **Coding**: Define `X` and `y` to set up the target (`median_house_value`) and features.

```
python
Copy code
X = df.drop('median_house_value', axis=1)  # Features
y = df['median_house_value']            # Target variable
```

Slide 5: Contribution

- **Coding**: Mention that each member contributed to different parts, such as data preprocessing, model selection, and evaluation. Example code might include quality checks and visualizations created by team members.

Slide 6: Methodology Overview

- **Coding**: Outline the steps of your methodology in code.

```
# Example of preprocessing and model training
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Slide 7: Technologies and Algorithms Used

- **Coding**: Import statements for the libraries used.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

Slide 8: Data Preprocessing

- **Coding**: Handle missing values, feature engineering, and scaling.

```
# Fill missing values in total_bedrooms
df['total_bedrooms'].fillna(df['total_bedrooms'].median(), inplace=True)

# Feature engineering
df['rooms_per_household'] = df['total_rooms'] / df['households']
df['bedrooms_per_room'] = df['total_bedrooms'] / df['total_rooms']

# Scaling example
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df.drop('median_house_value', axis=1))
```

Slide 9: Model Training and Evaluation

- **Coding**: Train and evaluate the models using Mean Squared Error (MSE) and R-squared (R^2).

```
from sklearn.metrics import mean_squared_error, r2_score

# Train a Random Forest model
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("MSE:", mse, "R-squared:", r2)
```

Slide 10: Results

- **Coding**: Show how to view feature importance in Random Forest.

```
# Feature importance from Random Forest
importances = model.feature_importances_
feature_names = X.columns
sorted(zip(importances, feature_names), reverse=True)
```

Slide 11: Visualizing Results

- **Coding**: Create a scatter plot of actual vs. predicted values for the Random Forest model.

```
plt.scatter(y_test, y_pred, alpha=0.3)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs. Predicted Housing Prices")
plt.show()
```

Slide 12: Conclusion

- **Coding**: Summarize your results based on the evaluation metrics and feature importance outputs.
-

Slide 13: Limitations

- **Coding**: Discuss any code adjustments or additional preprocessing steps that could be added to improve the dataset's quality and the model's generalizability.

Slide 14: Future Work

- **Coding**: Briefly mention that further hyperparameter tuning can be added or that you could implement additional models, like XGBoost.

```
from sklearn.model_selection import GridSearchCV
# Example GridSearch for hyperparameter tuning
param_grid = {'n_estimators': [50, 100, 150], 'max_depth': [10, 20, 30]}
grid_search = GridSearchCV(estimator=RandomForestRegressor(), param_grid=param_grid,
scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train, y_train)
```

Slide 15: Q&A

- **Coding**: Prepare to explain any specific code details if asked, and highlight how each part contributes to the overall prediction model.