```python
from google.colab import files
uploaded  = files.upload()
```

Choose Files housing.csv
housing.csv(text/csv) - 1423529 bytes, last modified: n/a - 100% done
Saving housing.csv to housing.csv

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error

# 1. Load Data (Ensure housing.csv is uploaded to Colab)
df = pd.read_csv('housing.csv')

# 2. Preprocessing
X = df.drop("median_house_value", axis=1)
y = df["median_house_value"]

numeric_features = X.select_dtypes(include=['float64', 'int64']).col
categorical_features = ["ocean_proximity"]

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=

# 3. Model Training and Comparison
models = [
    ('Linear (Baseline)', LinearRegression()),
```

```
        ('Ridge (Alpha=1.0)', Ridge(alpha=1.0)),
        ('Decision Tree', DecisionTreeRegressor(random_state=42)),
        ('Random Forest', RandomForestRegressor(n_estimators=50, random_
    ]

    results = []
    for name, m in models:
        pipe = Pipeline(steps=[('prepro', preprocessor), ('model', m)])
        pipe.fit(X_train, y_train)

        train_rmse = np.sqrt(mean_squared_error(y_train, pipe.predict(X_
        test_rmse = np.sqrt(mean_squared_error(y_test, pipe.predict(X_te
        test_mae = mean_absolute_error(y_test, pipe.predict(X_test))

        results.append([name, train_rmse, test_rmse, test_mae])

    # 4. Display Results Table
    comparison_df = pd.DataFrame(results, columns=['Model', 'RMSE Train'
    print("--- MODEL COMPARISON TABLE ---")
    print(comparison_df.to_string(index=False))
```

```
--- MODEL COMPARISON TABLE ---
            Model   RMSE Train    RMSE Test     MAE Test
Linear (Baseline) 68433.937367 70059.193339 50670.489236
Ridge (Alpha=1.0) 68434.995896 70066.021121 50676.922171
    Decision Tree     0.000000 69175.769189 43604.014293
    Random Forest 18396.283659 49073.147477 31930.318328
```

Bias-Variance Trade-off Observations on Model Performance: Underfitting (High Bias): The Linear and Ridge models show high RMSE on both training and test sets (≈ 70k). The models are too rigid to capture the complex spatial patterns of California housing. Overfitting (High Variance): The Decision Tree achieved a 0.00 Training RMSE, meaning it perfectly memorized the data. However, its Test RMSE is significantly higher, indicating it failed to generalize to new data. Generalization: The Random Forest achieved the lowest Test RMSE (≈ 49k), successfully balancing bias and variance by averaging multiple deep trees to reduce noise.

Real-World Issue: Non-Linearity & Outliers The dataset contains a "ceiling effect" where house values are capped at $500,001. This creates noisy labels that mislead the models. Furthermore, the relationship between location and price is highly non-linear, which explains why simple linear models performed poorly compared to ensemble methods.