# California Housing Dataset

## RAW FILE :

housing.csv

## DATA CLEANING :

### SOFTWARES USED FOR DATA CLEANING :

- EXEL ( rarely used for this  project )
- DATA WRANGLER
- JUPITER NOTEBOOK
- PYTHON LIBRARY PANDAS

## CODE :

```
# importing libraries
import numpy as np
import pandas as pd

# load data
data = pd.read_csv('housing.csv')

# to drop capped values
data = data[data['median_house_value'] < 500000]

# log transformation of target variable
data['log_median_house_value'] = np.log(data['median_house_value'])

# fill null values of total_bedrooms with mean
```

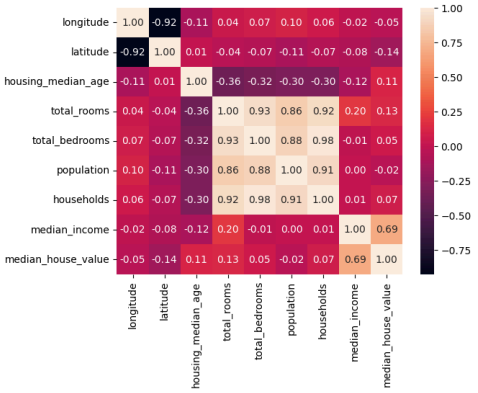
```
data['total_bedrooms'] = data['total_bedrooms'].fillna(data['total_bedroom
s'].mean())

# Standard data information
print(data.shape)
data.info()
data.head(3)
```

# FIRST LOOK INSIGHTS :

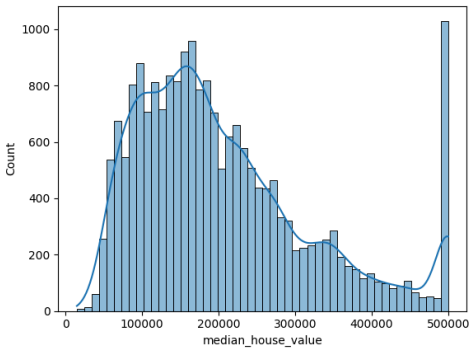
- Dataset contain  20640 rows and 10 columns.
- TARGET variable is `median_house_value` .
- Target variable contain capped value of 500000 which we had to drop  in order to get better insights.
- `median_house_value`  is right skewed to  treat this skewness log transformation is done.
- log transformed values are stored in `log_median_house_value` .
- The column `total_bedrooms` contain some null values which is treated by using mean of column `total_bedrooms` .

# VISUALIZATION CODE :

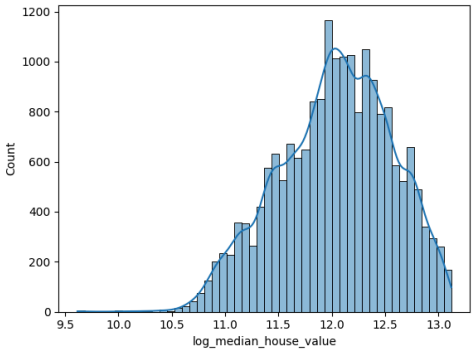
```
# heatmap of correlation between numerical features
numerical_data = data.select_dtypes(include=["float64", "int64"])


corr = numerical_data.corr()
sb.heatmap(corr, annot=True, fmt=".2f")
```



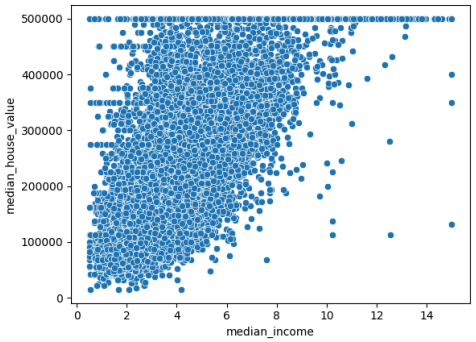

```
# histplot of median_house_value
sb.histplot(data['median_house_value'], bins=50, kde=True)
```



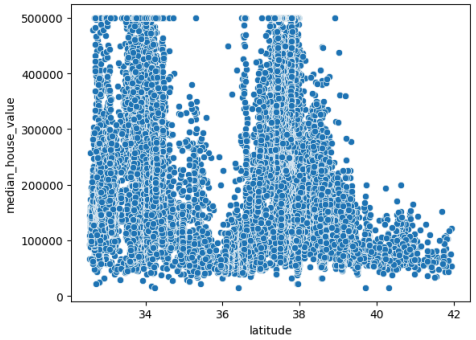

```
# histplot of log_median_house_value
sb.histplot(data['log_median_house_value'], bins=50, kde=True)
```




```
# scatterplot of median_income vs median_house_value
sb.scatterplot(x='median_income', y='median_house_value', data=data)
```




```
# scatterplot of latitude vs median_house_value
sb.scatterplot(x='latitude', y='median_house_value', data=data)
```




# INSIGHTS FROM GRAPHS :

- `longitude` has -0.05 correlation with target variable.
- `latitude` has -0.14 correlation with target variable.
- `housing_median_age` has 0.11 correlation with target variable.
- `total_rooms` has 0.13 correlation with target variable.
- `total_bedrooms` has 0.05correlation with target variable.
- `population` has -0.02 correlation with target variable.
- `households` has 0.07 correlation with target variable.
- `median_income` has highest correlation of 0.69 with target variable.
- `median_house_value` is right skewed.
- `median_income` vs `median_house_value` are in clustered form except caped value.

- In `latitude` vs `median_house_value` most of data is clustered around latitude values 34 and 38.

---

# MACHINE LEARNING MODELS :

1. LINEAR REGRESSION MODEL

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error , mean_squared_error

#load csv data
data = pd.read_csv('housing.csv')

#feature engennering
# to drop capped values
data = data[data['median_house_value'] < 500000]
# log transformation of target variable
data['log_median_house_value'] = np.log(data['median_house_value'])
# to drop ocean_proximity
data = data.drop(["ocean_proximity"], axis=1)
# fill null values of total_bedrooms with mean
data['total_bedrooms'] = data['total_bedrooms'].fillna(data['total_bedroom
s'].mean())

# Seprate feature from target
X = data.drop(["median_house_value" , "log_median_house_value"], axis=1)
Y = data["log_median_house_value"]

# train-test split
X_train , X_test , Y_train, Y_test = train_test_split(X, Y, test_size=0.2, rando
```

```
m_state=42)

# scale numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# train ridge regression
model = LinearRegression()
model.fit(X_train, Y_train)

# predictions
Y_train_pred = model.predict(X_train)
Y_test_pred = model.predict(X_test)

# calculate r2 score
r2 = r2_score(Y_test,  Y_test_pred)
mae = mean_absolute_error(Y_test, Y_test_pred)
mse = mean_squared_error(Y_test, Y_test_pred)
rmse = np.sqrt(mse)

# print values
print("R2 score : ", r2)
print("MAE : ", mae)
print("RMSE : ", rmse)
```

2. RIDGE REGRESSION :

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error , mean_squared_error

#load csv data
```

```
data = pd.read_csv('housing.csv')

#feature engennering
# to drop capped values
data = data[data['median_house_value'] < 500000]
# log transformation of target variable
data['log_median_house_value'] = np.log(data['median_house_value'])
# to drop ocean_proximity
data = data.drop(["ocean_proximity"], axis=1)
# fill null values of total_bedrooms with mean
data['total_bedrooms'] = data['total_bedrooms'].fillna(data['total_bedroom
s'].mean())

# Seprate feature from target
X = data.drop(["median_house_value" , "log_median_house_value"], axis=1)
Y = data["log_median_house_value"]

# train-test split
X_train , X_test , Y_train, Y_test = train_test_split(X, Y, test_size=0.2, rando
m_state=42)

# scale numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# train ridge regression
model = Ridge(alpha=0.01)
model.fit(X_train, Y_train)

# predictions
Y_train_pred = model.predict(X_train)
Y_test_pred = model.predict(X_test)

# calculate r2 score
r2 = r2_score(Y_test,  Y_test_pred)
mae = mean_absolute_error(Y_test, Y_test_pred)
mse = mean_squared_error(Y_test, Y_test_pred)
```

```
rmse = np.sqrt(mse)

# print values
print("R2 score : ", r2)
print("MAE : ", mae)
print("RMSE : ", rmse)
```

# OUTPUT :

1. RIDGE REGRESSION ( `log_median_house_value` )
   - R2 score : 0.6241129132796646
   - MAE : 0.10754611667733201
   - RMSE : 0.3279422459478681
2. Ridge REGRESSION ( `median_house_value` )
   - R2 score : 0.6002825208470519
   - MAE : 3832202018.691441
   - RMSE : 61904.781872577834
3. LINEAR REGRESSION ( `log_median_house_value` )
   - R2 score : 0.6241130060447148
   - MAE : 0.249335531703011
   - RMSE : 0.32794220548148617

# Insights :

Removing capped values improves realism but can reduce $R^2$
Log-transforming the target improves stability but changes metric interpretation

“Due to right-skewness and target censoring,
log transformation and capped-value removal were applied.
Ridge Regression achieved the most stable performance,
balancing bias–variance tradeoff better than tree-based models
on this dataset.”

# 🏠 California Housing Price Prediction

## 📌 Project Overview

This project focuses on predicting **median house values in California districts** using regression-based machine learning models.

The objective is to build a **robust, interpretable model** while carefully handling data skewness, multicollinearity, and target censoring.

---

## 📂 Dataset Description

- **Source:** California Housing Dataset
- **Rows:** ~20,640 (before preprocessing)
- **Features:** 8 numerical predictors
- **Target:** `median_house_value`

### Key Features

- `median_income`
- `housing_median_age`
- `total_rooms`
- `total_bedrooms`
- `population`
- `households`
- `latitude`
- `longitude`

---

## 🔍 Exploratory Data Analysis (EDA)

### Key Findings

- No missing values in most features
- `total_bedrooms` contained missing values → mean imputation applied
- Target variable is **right-skewed**

- Strong **price cap at $500,000**, causing artificial clustering
- `median_income` shows the strongest correlation with house price (~0.67)
- Several features are **highly correlated**, indicating multicollinearity

### Visual Insights

- Histogram revealed skewed distribution and capped values
- Scatter plots confirmed strong linear relationship between income and price
- Geographic clustering observed around major urban areas

---

## ⚙️ Data Preprocessing

### 1️⃣ Target Censoring

- Rows with `median_house_value == 500000` were removed
- Reason: These values represent a **capped ceiling**, not true prices

### 2️⃣ Target Transformation

- Log transformation applied:

```
log(median_house_value)
```

- Reduced skewness
- Improved stability for linear models

### 3️⃣ Missing Value Handling

- `total_bedrooms` filled using **mean imputation**

### 4️⃣ Feature Scaling

- StandardScaler applied
- Required for linear and regularized models

---

## 🤖 Models Implemented

### 🔹 Linear Regression

- Used as a baseline
- Performance limited by multicollinearity

---

### 🔹 Ridge Regression ✅ (Final Model)

- Addresses multicollinearity
- Provides stable coefficients
- Best generalization performance

**Configuration:**

- Alpha: 0.01
- Target: log-transformed
- Scaled features

**Performance:**

- **$R^2 \approx 0.62$**
- RMSE (log scale) ≈ 0.33

---

### 🔹 Random Forest Regression ❌

- Tested to capture non-linear relationships
- Underperformed due to:
  - Small dataset size
  - Predominantly linear relationships
  - Weak interaction effects

---

## 📊 Model Comparison

| Model | Target | $R^2$ |
|---|---|---|
| Linear Regression | log | ~0.60 |
| **Ridge Regression** | log | **~0.62** |
| Random Forest | raw | ~0.61 |

---

## 🧠 Key Learnings

- Model complexity does not guarantee better performance
- Regularization is critical when features are correlated
- Log transformation improves stability for skewed targets
- Tree-based models are not always optimal for small, linear datasets
- Proper EDA directly informs model choice

---

## ✅ Final Conclusion

> Ridge Regression with a log-transformed target provided the most stable and interpretable results for California housing price prediction, outperforming both basic linear regression and tree-based models.

---

## 🛠️ Tech Stack

- Python
- Pandas, NumPy
- Seaborn, Matplotlib
- Scikit-learn

---

## 🚀 Future Improvements

- RidgeCV for optimal alpha selection
- Polynomial and interaction features
- Residual diagnostics and assumption checks
- Gradient Boosting on engineered features

---

## 👋 Closing Note

This project demonstrates an **end-to-end machine learning workflow**:

EDA → Feature Engineering → Model Selection → Evaluation → Documentation