```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the dataset (replace 'your_dataset.csv' with your dataset file)
data = pd.read_csv('your_dataset.csv')

# Data preprocessing
# 1. Handle missing values if any
# data.dropna(inplace=True)  # Remove rows with missing values
# Alternatively, you can impute missing values

# 2. Feature selection and extraction
# Decide which features (columns) are relevant for the prediction
# X = data[['feature1', 'feature2', ...]]
# y = data['target_column']  # The column containing house prices

# 3. Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Initialize and train the model (in this case, Linear Regression)
model = LinearRegression()
model.fit(X_train, y_train)

# 5. Make predictions
y_pred = model.predict(X_test)

# 6. Evaluate the model
```

```
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# You can further fine-tune your model and try different algorithms for better results.
```

PREDICTING HOUSE PRICES USING MACHINE LEARNING

Introduction

Welcome to the presentation on *Forecasting House Prices with Machine Learning: An Advanced Predictive Modeling Approach*. In this presentation, we will explore how machine learning techniques can be applied to accurately predict house prices. We will discuss the importance of accurate house price predictions for various stakeholders in the real estate industry.

Objective

Our objective is to demonstrate how advanced predictive modeling techniques can be used to forecast house prices with high precision. We will delve into the details of different machine learning algorithms and their application in the real estate domain. By the end of this presentation, you will have a clear understanding of how machine learning can revolutionize house price predictions.

Machine Learning Algorithms for House Price Predictions

We will explore various machine learning algorithms suitable for house price predictions. This includes linear regression, decision trees, random forests, support vector machines, and neural networks. We will discuss the strengths and weaknesses of each algorithm and their applicability in the context of house price predictions. Understanding these algorithms will enable us to choose the best model for our predictive modeling approach.

Importance of House Price Predictions

Accurate house price predictions are crucial for various stakeholders, including homebuyers, sellers, real estate agents, and investors. *Homebuyers* rely on accurate predictions to make informed decisions about their investments. *Sellers* need to price their properties competitively. *Real estate agents* can provide better guidance to their clients. *Investors* can identify profitable opportunities. Machine learning can enhance the accuracy of these predictions, benefiting all stakeholders.

DESIGN AND DEVELOPMENT

Since this project mainly integrates software features rather than hardware ones, the emphasis is primarily on code development as follows-

OVERVIEW OF STEPS

Step 1

- Import required libraries
- Import data from dataset
- Get some information about dataset.

Run Markdown

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import systemcheck
from sklearn.metrics import mean_absolute_percentage_error
```

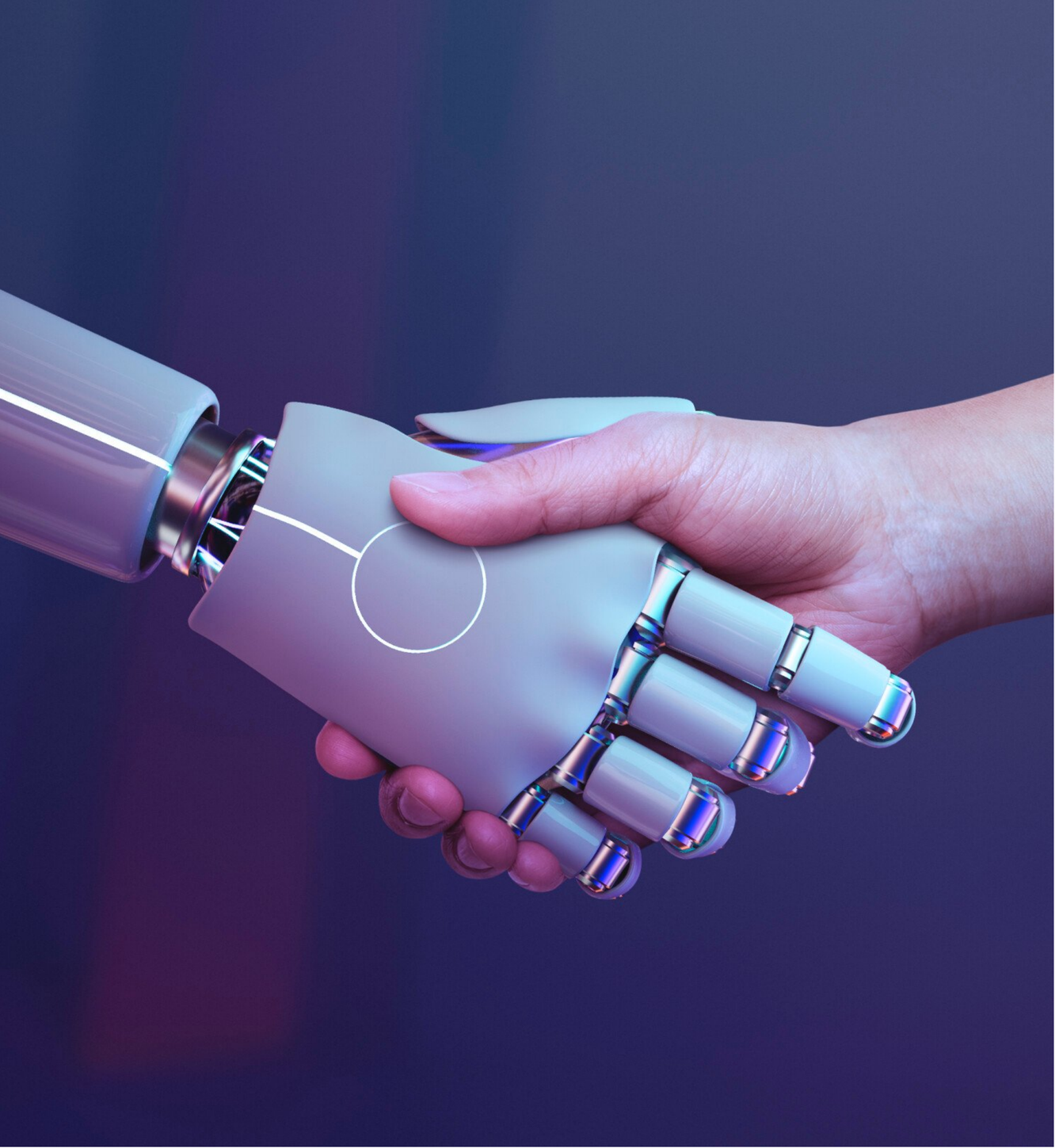
In [2]:
```
data = pd.read_csv("housing.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

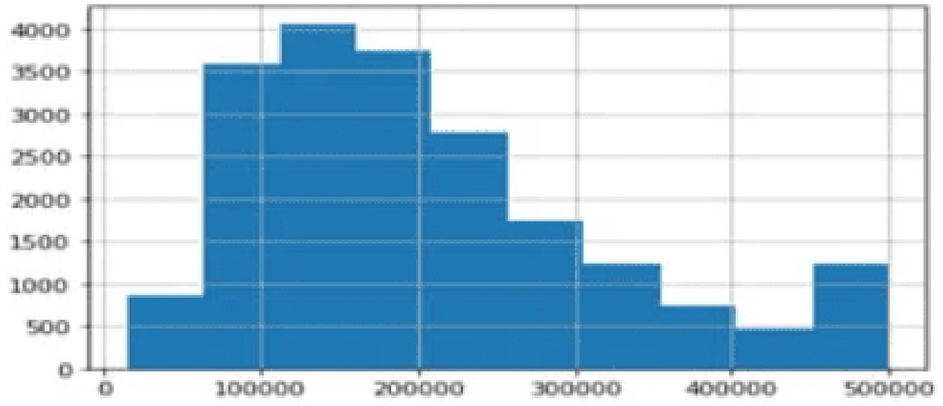
In [4]:
```
data[data["total_rooms"]<1000].head()
```



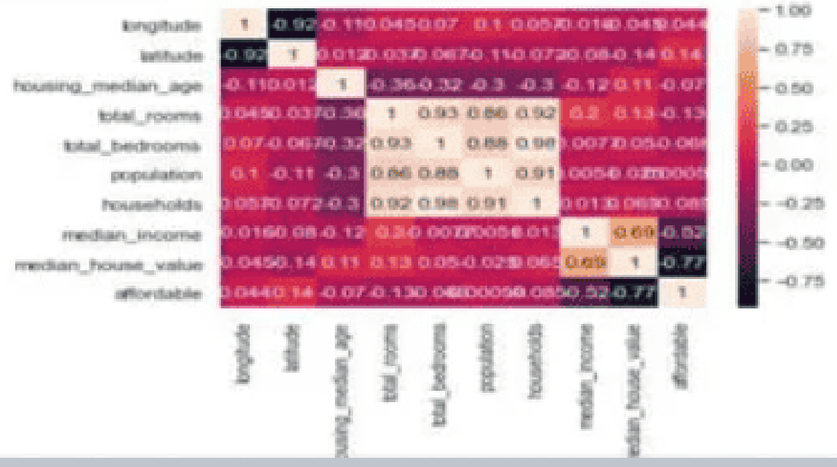
Step 2-

- Visualizing the data in terms of house prices, affordability, closeness to ocean, number of bedrooms etc.
- First we display a histogram for column ‘median_house_value’ from the dataset
- Next we display a Heatmap of the dataset. A heatmap() method contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade.
- Next we display a Distplot() for columns ‘median_house_value’, ‘affordable’, and ‘ocean_proximity’. A distplot() method lets you show a histogram with a line on it. It plots a univariate distribution of observations.

```
In [10]: data["median_house_value"].hist()
         plt.show()
```

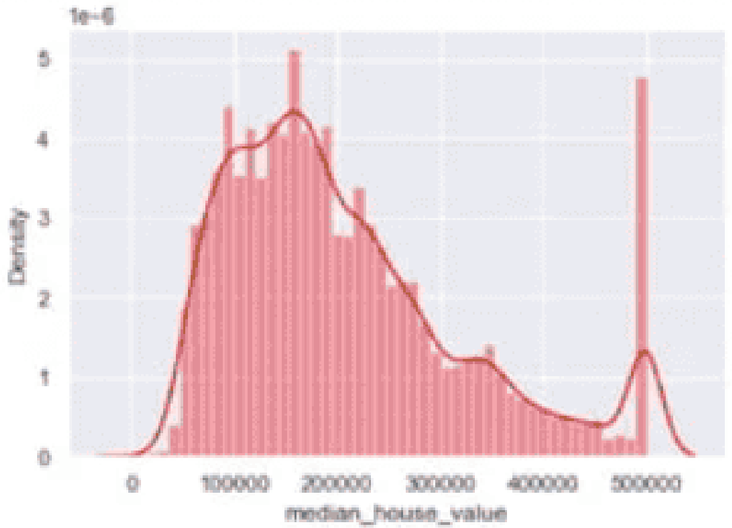



```
In [11]: sns.set()
         sns.heatmap(data.corr(), annot = True)
         plt.figure(figsize = (30,25))
         plt.tight_layout()
         plt.show()
```



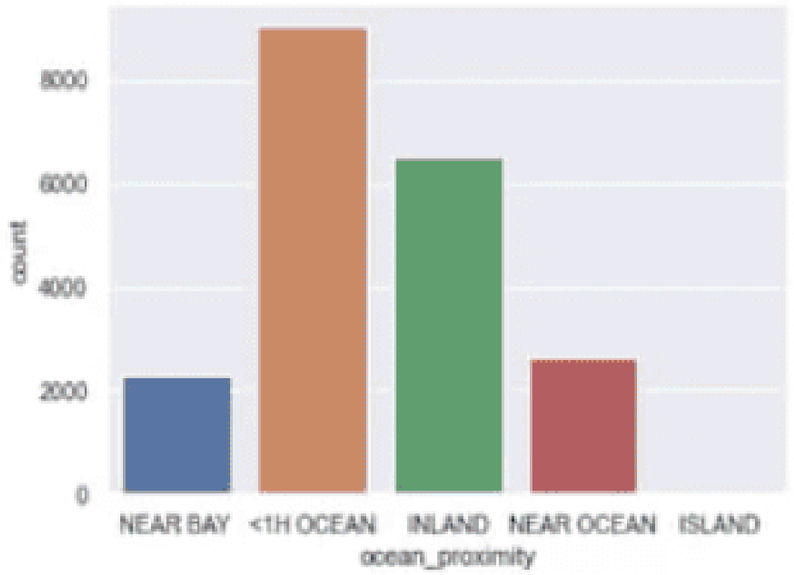
In [12]:
```
sns.set_color_codes(palette = "bright")
sns.distplot(data["median_house_value"], color = "r")
plt.show()
```

```
C:\Users\sreek\AppData\Local\Programs\Python\Python38\lib\site-packages\
` is a deprecated function and will be removed in a future version. Plea
vel function with similar flexibility) or `histplot` (an axes-level func
  warnings.warn(msg, FutureWarning)
```



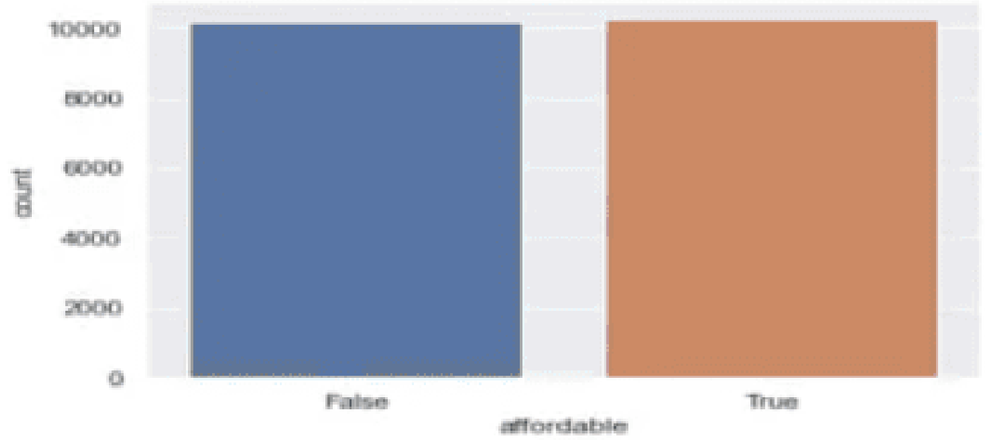
- Next we display a Countplot() for the column ‘ocean_proximity’. A countplot() method is used to show the counts of observations in each categorical bin using bars.
- Next we concatenate the values of dataset with values of ‘ocean_proximity’ column according to the different levels of distance between the ocean and housing area.
- Next we view the countplot() of ‘affordable’ column, convert it into string and replace string type “true, false” values by int values 0,1 to represent if the house is affordable or not.

In [16]:
```
sns.countplot(x=data["ocean_proximity"])
plt.show()
```



In [17]:
```
data = pd.concat([data,pd.get_dummies(data["ocean_proximity"], prefix="ocean_proximity")],axis=1)
data.drop(columns=["ocean_proximity"],inplace=True)
data
```

In [18]:
```
sns.countplot(x=data["affordable"])
plt.show()
#balanced data
```



In [20]:
```
data["affordable"] = data["affordable"].astype(str)
```

In [21]:
```
data["affordable"].replace(["True","False"],[1,0], inplace = True)
```

In [22]:
```
data.head()
```

Out[22]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	affordable	ocean_pro
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	0	

- Next we use the info() function on the dataset which basically shows the datatypes used for different columns.

```
In [23]: data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 20433 entries, 0 to 20639
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   longitude                   20433 non-null  float64
 1   latitude                    20433 non-null  float64
 2   housing_median_age          20433 non-null  float64
 3   total_rooms                 20433 non-null  float64
 4   total_bedrooms              20433 non-null  float64
 5   population                  20433 non-null  float64
 6   households                  20433 non-null  float64
 7   median_income               20433 non-null  float64
 8   median_house_value          20433 non-null  float64
 9   affordable                  20433 non-null  int64  
 10  ocean_proximity_<1H OCEAN   20433 non-null  uint8  
 11  ocean_proximity_INLAND      20433 non-null  uint8  
 12  ocean_proximity_ISLAND      20433 non-null  uint8  
 13  ocean_proximity_NEAR BAY    20433 non-null  uint8  
 14  ocean_proximity_NEAR OCEAN  20433 non-null  uint8  
dtypes: float64(9), int64(1), uint8(5)
memory usage: 2.3 MB
```

Step 3-

- Detecting and removing outliers using ‘IsolationForest’ algorithm.
- Isolation forest is an unsupervised learning algorithm for anomaly detection that works on the principle of isolating anomalies. It isolates the outliers by randomly selecting a feature from the given set of features and then randomly selecting a split value between the maximum and minimum values of the selected feature.
- In our dataset after running the algorithm we get around 7170 outliers in the form of negative values(-1), which we remove from the data, i.e., we delete 7170 rows from around 20,000 rows in the dataset.

In [24]:
```
from sklearn.ensemble import IsolationForest
clf = IsolationForest(max_samples=100, random_state = 0, contamination= 'auto')
preds = clf.fit_predict(data)
preds
```

Out[24]:
```
array([-1, -1, -1, ...,  1,  1,  1])
```

In [25]:
```
print("number of outliers: ",list(preds).count(-1))
```

```
number of outliers:  7170
```

In [26]:
```
for i,j in list(zip(range(len(preds)),data.index)):
    if preds[i] == -1:
        data.drop([j],inplace=True)
```

In [27]:
```
data
```

Out[27]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	affordable	ocean
10	-122.26	37.85	52.0	2202.0	434.0	910.0	402.0	3.2031	281500.0	0	
21	-122.27	37.85	42.0	1639.0	367.0	929.0	366.0	1.7135	159800.0	1	
90	-122.27	37.80	16.0	994.0	392.0	800.0	362.0	2.0938	162500.0	1	
93	-122.27	37.79	27.0	1055.0	347.0	718.0	302.0	2.6354	187500.0	0	
106	-122.24	37.81	52.0	2026.0	482.0	709.0	456.0	3.2727	268500.0	0	
...	...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	1	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	1	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	1	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	1	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	1	

13263 rows × 15 columns

Step 4-

- Splitting into training and testing data.
- We use the Train set to make the algorithm learn the data's behavior and then check the accuracy of our model on the Test set.
- Divide the data into two values-

✓Features (X): The columns that are inserted into our model will be used to make predictions.

✓Prediction (y): Target variable that will be predicted by the features.

- After defining values of x, y we import train and test functions from scikit learn library.

Case Study: Real Estate Market Analysis

In this section, we will present a real-world case study on real estate market analysis using machine learning. We will showcase how the techniques discussed in this presentation can be applied to a specific market and provide valuable insights. The case study will demonstrate the practical application and effectiveness of our advanced predictive modeling approach.

Limitations and Future Directions

While machine learning techniques offer promising results in house price predictions, there are limitations to consider. We will discuss the challenges associated with data availability, model interpretability, and potential biases. Additionally, we will explore future directions, including the integration of external data sources, incorporation of domain knowledge, and advancements in deep learning for more accurate predictions.

Conclusion

In this presentation, we have explored the application of machine learning in forecasting house prices. We have discussed the importance of accurate predictions for various stakeholders and the challenges associated with house price predictions. By leveraging advanced predictive modeling techniques, we can overcome these challenges and achieve high-precision predictions. Machine learning has the potential to revolutionize the real estate industry by providing valuable insights and improving decision-making processes.


Thank you