A SHORT-TERM INTERNSHIP REPORT ON

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

BY

BHANU PRASAD KURAMDASU

III Bsc Data Science

Under the Esteemed Guidance of

Mr. G.V.S.S PRASANTH SIR

Tutor of Artificial Intelligence & Machine Learning

ADITYA DEGREE COLLEGE,

TUNI.

(Affiliated to Adikavinannaya University)

TUNI-533401, Kakinada District, ANDHRA PRADESH

2022-2025

ADITYA DEGREE COLLEGE

DECLARATION BY THE STUDENT

I hereby declare that the work described in this Short-term Internship, entitled "**Artificial Intelligence & Machine Learning**" which is being submitted by me in partial fulfilment of the requirements for the award of degree of **Bachelor of Computer Science** from the Department of Bachelor of Computer Science to Aditya Degree College, Tuni under the guidance of **Mr. G.V.S.S PRASANTH Sir** tutor of **Artificial Intelligence & Machine Learning** in Aditya Degree College , Tuni.

Place: TUNI **(BHANU PRASAD KURAMDASU)**

Date:

ADITYA DEGREE COLLEGE

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the Short Term Internship entitled, “**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING”,** that is being submitted by **BHANU PRASAD KURAMDASU** bearing **221177156171** of **III DSSTCS**, which is being submitted by us in partial fulfilment of the

requirements for the award of degree of **Bachelor of Computer Science** from the Department of Bachelor of Computer Applications to Aditya Degree College, bonified work carried out by him under my guidance and Supervision.

(Mr. G.V.S.S PRASANTH SIR)

ACKNOWLEDGEMENT

No endeavour is completed without the valuable support of others. I would like to take this opportunity to extend my sincere gratitude to all those who have contributed to the successful completion of this Short-Term Internship Project Report.

I express my deep sense of gratitude to **Mrs. M. DEEPTHI Principal**, for her Efforts and for giving us permission for carrying out this Long-Term Internship.

I feel deeply honoured in expressing my sincere thanks to Mr. G.V.S.S Prashanth Sir tutor of U-Learn Visakhapatnam for making the resources available at right time and providing valuable insights leading to the successful completion of my short-Term Internship Project Report.

Finally, I thank all the faculty members of our department who contributed their valuable suggestions in completion of Short-Term Internship report and I also put my sincere thanks to My Parents who stood with me during the whole Short-Term Internship.

(K. BHANU PRASAD)

INTRODUCTION

Exploratory Data Analysis (EDA) is a critical step in the data science process that involves summarizing the main characteristics of a dataset, often using visual methods. When applied to Housing Prices Prediction data, EDA helps in understanding the underlying patterns, relationships, and trends within the data, providing valuable insights for business decision-making.

Objectives of EDA in Housing Prices Prediction :-

- Objective: Predicting the median housing price for given block.
- Problem Type: Supervise & Regression

The dataset contains information about houses in California district, obtained from 1990 California census. There are around 20000 records along with 10 features in the dataset. Feature names are self explanatory: longitude, latitude, housing_median_age, total_rooms, total_bedrooms, population, households, median_income, median_house_value, ocean_proximity

Important to remember that the each record in dataset is not about the house but it is about the block.

1. longitude: A measure of how far west a house is; a higher value is farther west
2. latitude: A measure of how far north a house is; a higher value is farther north
3. housingMedianAge: Median age of a house within a block; a lower number is a newer building
4. totalRooms: Total number of rooms within a block.
5. totalBedrooms: Total number of bedrooms within a block.
6. population: Total number of people residing within a block.

Benefits Of EDA in California Housing Prices :-

Exploratory Data Analysis (EDA) plays a crucial role in extracting meaningful insights and understanding the California housing prices dataset. Here are some specific benefits of EDA in this context:

1. **Identifying Trends and Patterns**: EDA allows analysts to uncover trends in housing prices across different regions of California over time. By visualizing data through plots like time series graphs or spatial maps, analysts can identify patterns such as seasonal fluctuations, long-term trends, and cyclical variations.

2. **Understanding Relationships**: EDA helps in exploring relationships between various variables such as median house prices, income levels, population density,

and housing quality metrics. Through correlation analysis and scatter plots, analysts can identify how these factors interact and influence each other.

3. **Detecting Outliers and Anomalies**: EDA techniques such as box plots and histograms enable analysts to detect outliers in dataset. Outliers may represent unique cases or errors in data entry but can also reveal interesting insights about extreme housing prices or unusual market conditions.

4. **Assessing Data Quality**: EDA allows analysts to assess the quality and completeness of the dataset. By examining missing values, data distributions, and data consistency, analysts can determine if further data cleaning or preprocessing steps are necessary before proceeding with more advanced analyses.

5.**Supporting Feature Selection**: In predictive modelling tasks, EDA helps in selecting relevant features (variables) that are most predictive of housing prices. Techniques like feature importance ranking or principal component analysis (PCA) can aid in identifying key predictors from a potentially large set of variables.

6.**Informing Hypothesis Generation**: EDA often sparks hypotheses about the underlying factors influencing housing prices in California. For example, by observing a correlation between rising incomes and increasing housing prices in certain regions, analysts may hypothesize about the impact of economic growth on real estate values.

5. **Facilitating Stakeholder Communication**: EDA results are often presented visually through charts, graphs, and summary statistics, making complex data more accessible to stakeholders. Clear visualizations enable policymakers, real estate developers, and investors to grasp key insights quickly and make informed decisions.

In essence, EDA serves as a foundational step in the analysis of the California housing prices dataset, providing a comprehensive understanding of the dataset's structure, characteristics, and relationships. It empowers analysts to extract actionable insights that drive informed decision-making and policy formulation in the dynamic real estate market of California.

LEARNING OUTCOME OF SHORT TERM INTERNSHIP

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

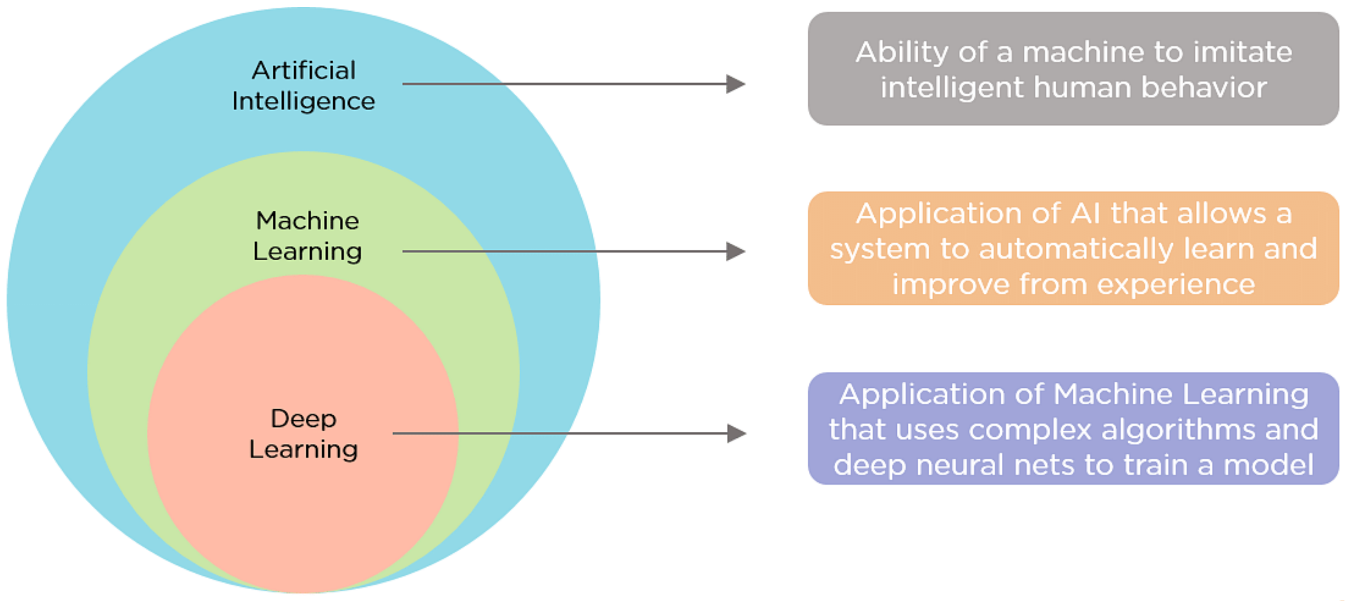
INTRODUCTION

What is AI?

- Ai is an branch of computer science it is the simulation of humanintelligence process by machines is called AI
- Human intelligence which performs machines is called AI

In AI how many subsets are there?

- Machine learning
- Deep learning


Artificial Intelligence
Ability of a machine to imitate intelligent human behavior
Machine Learning
Application of AI that allows a system to automatically learn and improve from experience
Deep Learning
Application of Machine Learning that uses complex algorithms and deep neural nets to train a model

Sub fields

- Artificial neural networks
- Cognative computing
- Natural language processing

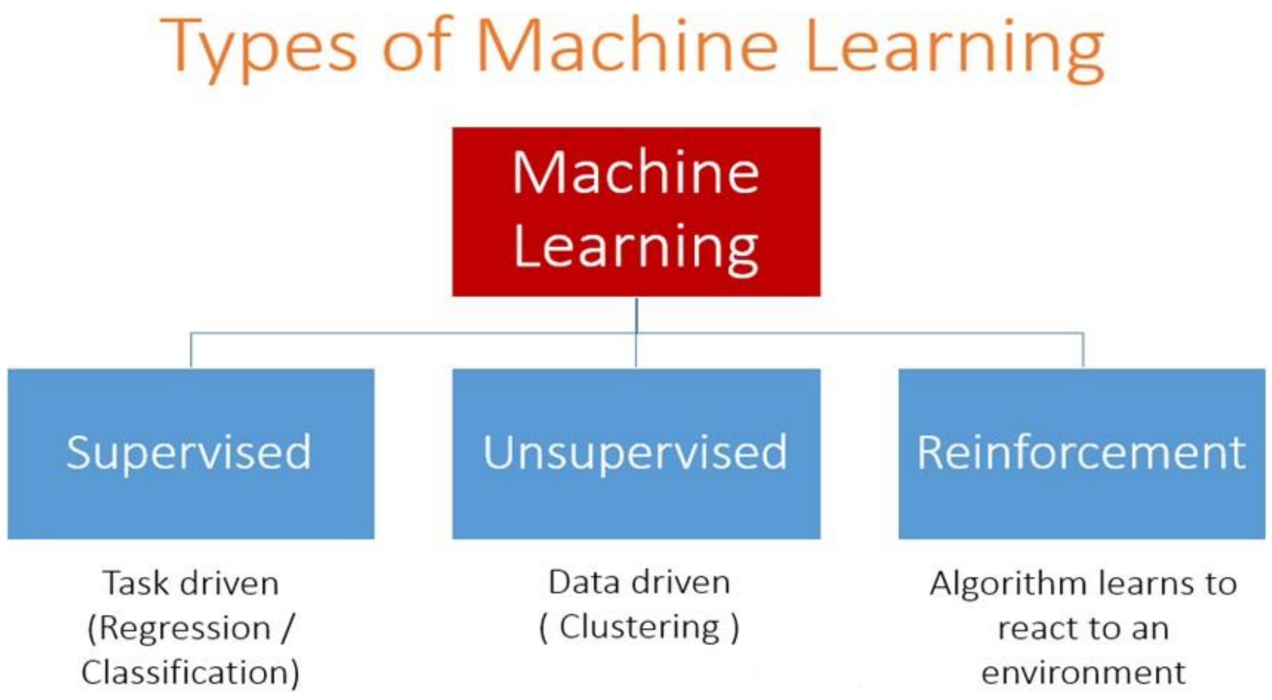
What is Machine learning?

- Its a subset of ai which focus on the use of data and algorithmsimitate the way that human learn and fradually increasing its accuracy
- It learns from data & solve the problemsTypes of

machine learning:

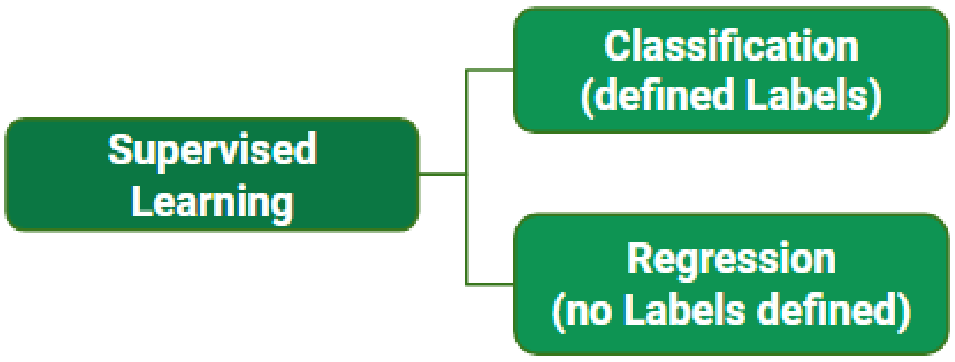
There are 3types of ML

- Supervised learning - its an labelled data or structered data
- Unsupervised learning - its an unlabelled data or unstructered data
- Reinforcement learning -it uses both structured data and unstructureddata


Types of Machine Learning
Machine Learning
Supervised
Unsupervised
Reinforcement
Task driven (Regression / Classification)
Data driven (Clustering)
Algorithm learns to react to an environment

SUPERVISERED LEARNING:

- Classification
- Regression


Supervised Learning
Classification (defined Labels)
Regression (no Labels defined)

Classification:

- KNN
- Support vector Machine
- Decision Tree
- Random Forest
- Navie Bays
- Neural Network
- Stochastic Gradient Descent

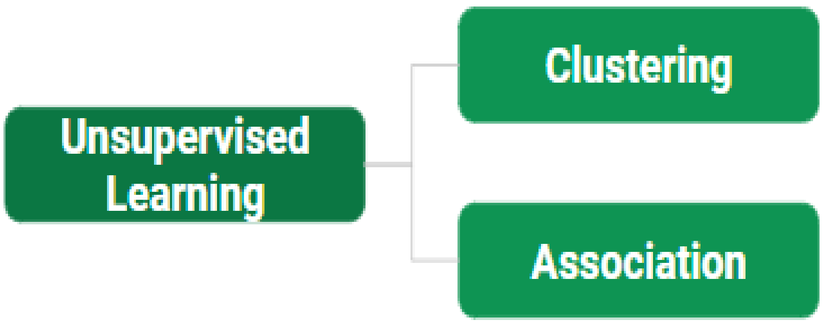
Regression:

- Linear regression
- Logistic Regression

- Polynomial Regression

UNSUPERVISED LEARNING:

- Clustering
- Association



Clustering:

- Principal Component Analysis
- Hierarchical Clustering
- Singular Value Decomposition
- Independent Component Analysis
- K-Means Clustering

Reinforcement Learning:

3 Ways of implementation

- Model based
- Policy based
- Value based

Models:

- Q-learning
- Markov Models

Programming Languages used in ML:

- Python
- R Language

Python:

Python is one of the easiest yet most useful programming languages which is widely used in the software industry. People use Python for Competitive Programming, Web Development, and creating software. Due to its easiest syntax, it is recommended for beginners who are newto the software engineering field. Its demand is growing at a very rapid pace due to its vast use cases in Modern Technological fields like Data Science, Machine learning, and Automation Tasks.

Mathematics:

- Linear Algebra
- Calculas
- Probability

Data Bases:

To store the structured or unstructured data in database to accessthe data by using Sql queries (structured query language)

- MangoDB
- My SQL

Visualization Tools:

- Qlick Sense
- Tableau

- PowerBI

Why we used this bi tools?

For data visulatation purposeBar Graphs

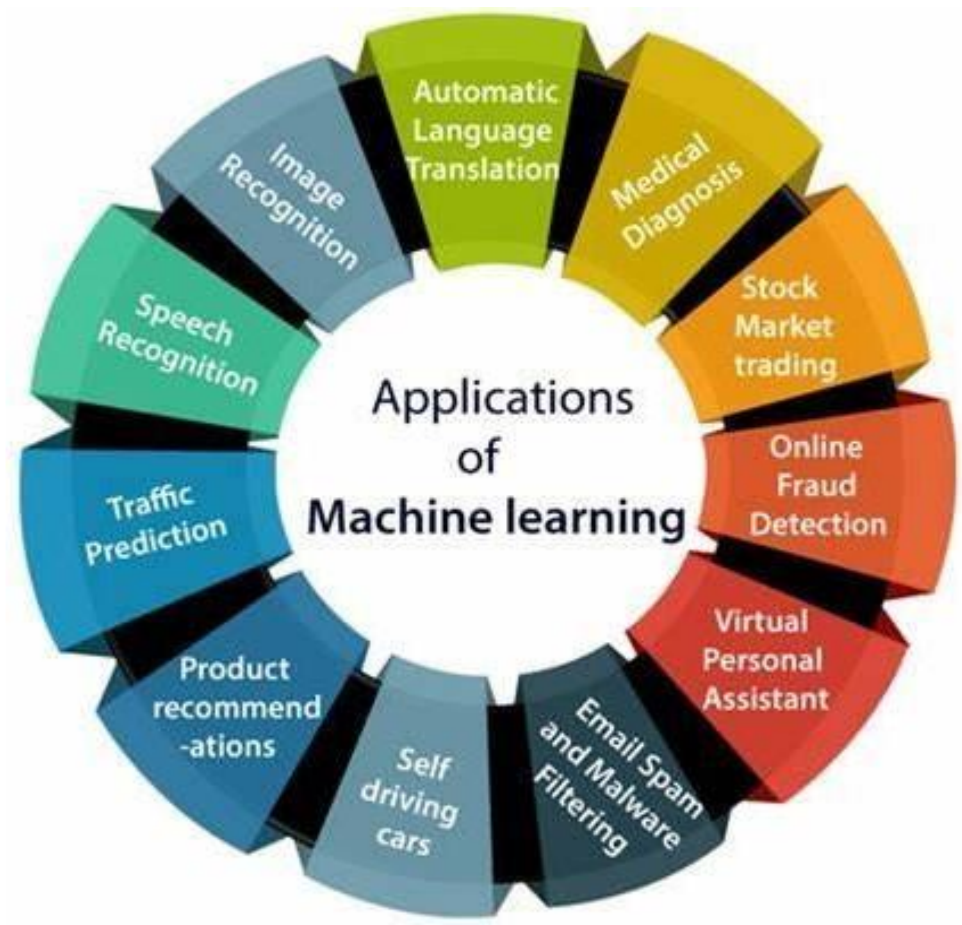
Pie charts
Histograms
Box plots
Scatterplot
Line charts

Steps required to train AI And ML models:

- Gathering Data
- Data prepration
- Choosing A model
- Training
- Evaluation
- Parameter tuning
- Prediction

<u>Applications of Machine Learning:</u>

- Self-Driving Cars
- Chatbot
- Image registration
- Speech Recognition
- Stock Market trading
- Email Spam Filtering
- Online fraud Detection


Applications of Machine learning
Automatic Language Translation
Medical Diagnosis
Stock Market trading
Online Fraud Detection
Virtual Personal Assistant
Email Spam and Malware Filtering
Self driving cars
Product recommend -ations
Traffic Prediction
Speech Recognition
Image Recognition

<u>AI tools we used in our daily life:</u>

1. **Chatbots**

A prominent example of this is **ChatGPT**. People started out using this chatbot justanother online companion. However, you'll be surprised to know that, ChatGPT is actually an **artificial intelligence-powered chatbot.**

2. **<u>Microsoft Bing</u>**

While Google has always been the go-to search engine for almost everyone, Microsoft has now revamped Bing with artificial intelligence. The new AI Bing has been specially created to give the search engine the power to intelligently give nuanced responses by its AI. However, Bing also benefits from the new Chat mode.

3. Smart Compose, Quick Reply, and Grammar Check

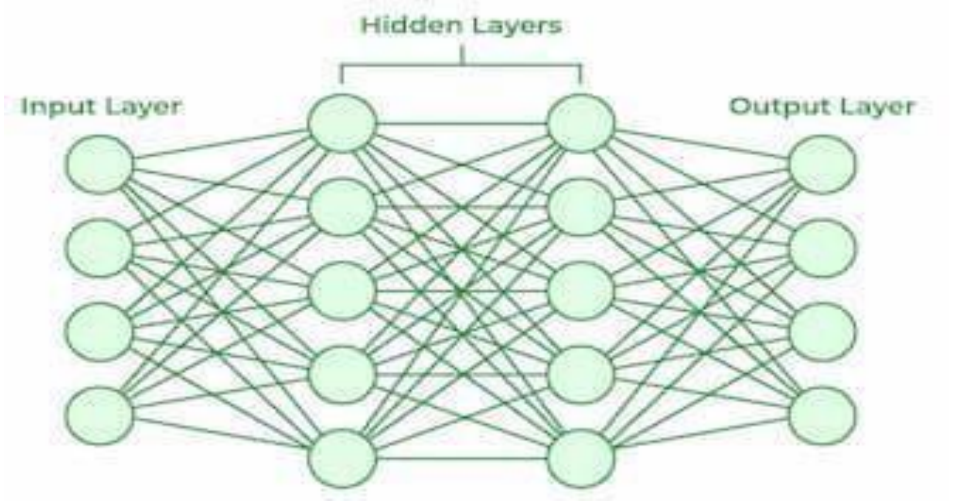
If you use Gmail then you might have noticed a feature called Smart Compose. It suggests complete sentences based on the preceding line that you have written. It uses Artificial Intelligence to quickly compose your **email drafts with contextual accuracy** and correct grammar. I use it quite often and believe me, it’s pretty helpful.There could be no better example of AI making life better and saving time on the other hand.

4. Google Lens and OCR

Google Lens is another Google service that is built on AI and has some great technology for fast and accurate optical recognition. It allows you to search for anything through images. Just point the camera to a shoe or a plant or   a text, and it can detect the type of subject and will provide precise information onthat thing in just a few seconds

What is neural Network?

Neural Networks are computational models that mimic the complex functions of thehuman brain. The neural networks consist of interconnected nodes or neurons that process and learn from data, enabling tasks such as pattern recognition and decision making in machine learning. The article explores more about neural networks, their working, architecture and more.



Input Layer: Each feature in the input layer is represented by a node on the network, which receives input data.

Weights and Connections: The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.

Hidden Layers: Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.

Output: The final result is produced by repeating the process until the output layer is reached.

Back propagation:

- In machine learning, backpropagation is an effective algorithm used to trainartificial neural networks, especially in feed-forward neural networks.
- Backpropagation is an iterative algorithm, that helps to minimize the cost function by determining which weights and biases should be adjusted. During every epoch, the model learns by adapting the weights and biases tominimize the loss by moving down toward the gradient of the error. Thus, it involves the two most popular optimization algorithms, such as gradient descent or stochastic gradient descent.

Objection Detection:

Object detection is a technique that uses neural networks to localize and classify objects in images. This computer vision task has a wide range of applications,from medical imaging to self-driving cars.

Applications:

- Object reg
- Facial Recg
- Video Tracking
- Moment Detection
- Self Driving Cars
- Animal Detection
- Robotics

Steps involved in objection detection:

- Image preprocessing - resizing and normalization of image
- Feature extraction - it classifies the image
- Object localization - it locates the object
- Object classification - it identifies what type of object in image
-Post processing - it refining and eliminating the duplicate detections.

Gan - generative adversarial networks:

-Gan was introduced by LAN goodfellow in 2014.
-Gan are algorithmic architecture that uses two neural networks pitting oneagainst to other to generate new data that passes through the real data.

Applications:

- To generate photorealistic images
- Change facial expressions
- Create computer game scenes
- Visualize designs
- Create artwork

Diff b/w original image and generated images:

-Brightness
-Thickness
-Color
-Background
-Saturation
-unrealistic elements
-Quality
-Clarity
-Size
-Features

DEEP DREAM:

- Deep dream is one of the application of deep learning in computer vision.
- In this deep dream concept we used deep neural networks
- We are using CNN algorithm to find image patterns in images

- Deep dream software original image using deep CNN named as inception

PROCESS:

- If we take any image the deepdream will identify faces and there patterns inimage by using deep CNN algorithm to modify the images.
- once we trained this algorithm its reverse process takes place tochangethe image patterns.
- This can be visualizations to understand the emergent structure ofneuralnetwork and basis for the deep dream concept

DEEP FAKE:

It is digitally alterated image, video, or audio that replaces one person facewithanother person face is called deep fake

Algorithms:

- Deep CNN
- GAN Model
- VGG16 and VGG19 - visual geometrical graphs

Training:

Jupyter notebook:

- required libraries
- importing datasets
- selecting input image
- targeted image
- training algorithm
- output

DATA AUGMANTATION:

Data augmentation uses pre-existing data to create new data samples that canimprove model optimization and generalizability.

- Data Augmantation is data analysis are techniques used to increasedthe amount of data by adding or slightly modifying copies in already existing dataor newly created synthetic data is called data augmantation.
- It is a set of techniques to artificially increase the amount of datagenerating new data points from existing data

WHY ITS IMPORTANT?

- It includes making small changes to data or using deep learning models togenerate new data points.
- it is useful to improve the performance and outcomes of machine learningmodels by forming new and different examples to train datasets.

STEPS:

- Input data that feed to the data Augmantation pipeline
- The data Augmantation pipeline by sequential steps with different Augmantations
- Tf1-rotation
- Tf2-greyscaleto rgbtf3-blur
- Tfn-flip
- The image is feed through the pipeline and processed through each step withdifferent probablity
- After image is processed the human expert randomaly verifies the augmantedresults and passes the feedback to the system
- After human expert verification the augmanted data is ready to train the aitraining process

FOR IMAGE CLASSIFICATION AND SEGMENTATION:

Random
Rotatingre
Scalling
Vertical or horizontal flipping

ranslation
Cropping
zooming

PARAMETER SHARING AND TYPING:

It is an convolutional neural network model which is used to share the weightsequally in neural networks is called parameter sharing and typing

It is an deep learning application

Parameter sharing is the method of sharing weights by all neurons in aparticular feature map

ENSEMBLE METHODS:

- Ensemble methods are techniques that aim to improving the results inmodelsby combining multiple model instead of using single model.
- The combined models increase the accuracy of the results.
- The most popular ensemble methods are bagging boosting

SENSEMBLE METHODS:

- BAGGING
- BOOSTING

BOOTSTACKING:

- The argumanted data is trained with multiple models in AI process the accuracyresults is more
- Ensemble methods are ideal for regression and classification where theyreduce bias and variance to accuracy of models

BAYES THEORM:

A bayes theorm finds the probability of an event occuring given the probabilityofanother event that has already occurs is called bayes theorm.

P(A/B) = P(B/A) P(A) / P(B)

Ex: if we toss a coin the probabilites = heads and tailstheprobability of getting heads = 50%

The probability of getting tails = 50%
The occuring probability = 100%

We want to know that having alley when the text says
P(Y) = 1% * 80% + 99% +10%
= 10.7%

P(A/Y) = P(A) P(Y/A) / P(Y)
= 1% + 80% / 10.7%
= 7%

LSTM - LONG SHORT TERM MEMORY:

- LSTM network is a type of recurrent neural network architecture that is designed by the problem of vanishing expanding gradients in traditionalRNN.
- LSTM are widely used in deeplearning for sequential data analyzingsuch asspeech recoginization, NLP, & time series analysis
- The architecture of lSTM network is similar to that trend RNN, but it includesmemory cell & three gates

▪ Input gates
▪ Forget gates
▪ Output gates

Restricted boltzman machine:

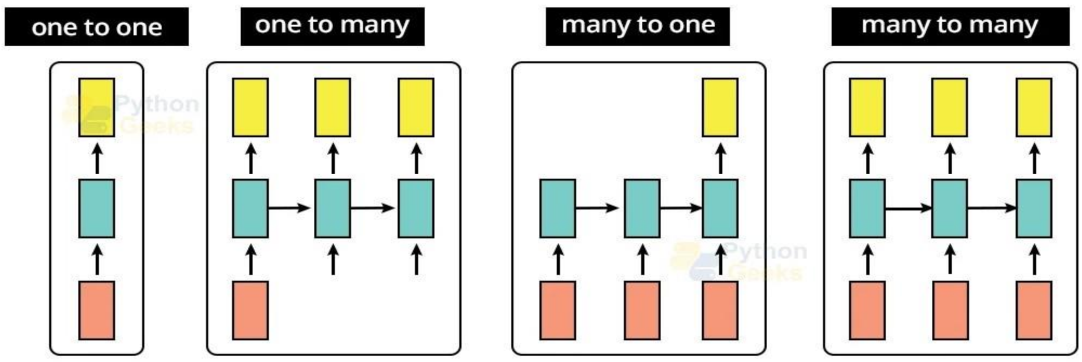
- A RBM is a type of artificial neural networks that commonly used in unsupervised machine learning tasks, such as dimensionally reduction, featurelearning, and collebrative learning to represent the probability distribution
- The RBM consist of 2 layers visible layer and hidden layer
- The visible layers represents the input data and the hidden layers represents aset of latent variables that captures the under lying because they dont have common connections

RNN-- recurrent neural networks

- In RNN we have seprate and indipendent input and output layers
- Which we inefficient the dealing with sequencial data
- Hence a new neural network called rnn was introdued to store of previousoutputs in the internal memory
- These results are then fed into the neural network as input this allows it toused in applications like pattern detection speech recoginization, nlp, timeseries predection
- RNN has hidden layers that acts memory locations to store the output of alayer in a loop

There are 4types are there in recurrent neural network

- One to one
- One to many
- Many to one
- Many to many



1. one to one:

- In rnn is one to one which allows a single input & single output
- It has fired input and output sizes and acts as a traditional neural networks applications

2. One to many:
   - One to many is a type of RNN that gives multiple outputs which we given singleinput.
   - It table a fixed size and give a sequential of data inputs and themain applications are found in music generation and image capturing

3. Many to one:
   - Many-to-one is used when a single output is required from multiple inputs insequence
   - It takes a sequence of inputs to display fixed output

4. Many to Many
   - It is used to generate the sequence of output data from a sequence ofinputdata

Auto encoders:

- An auto encoder AI a type of neural network architecture that is used inunsupervised learning
- The main goal of auto encoder is to learn a component representation of theoriginal data

Auto encoders consists of two parts
- Encode
- Decode

Types of auto encoders:

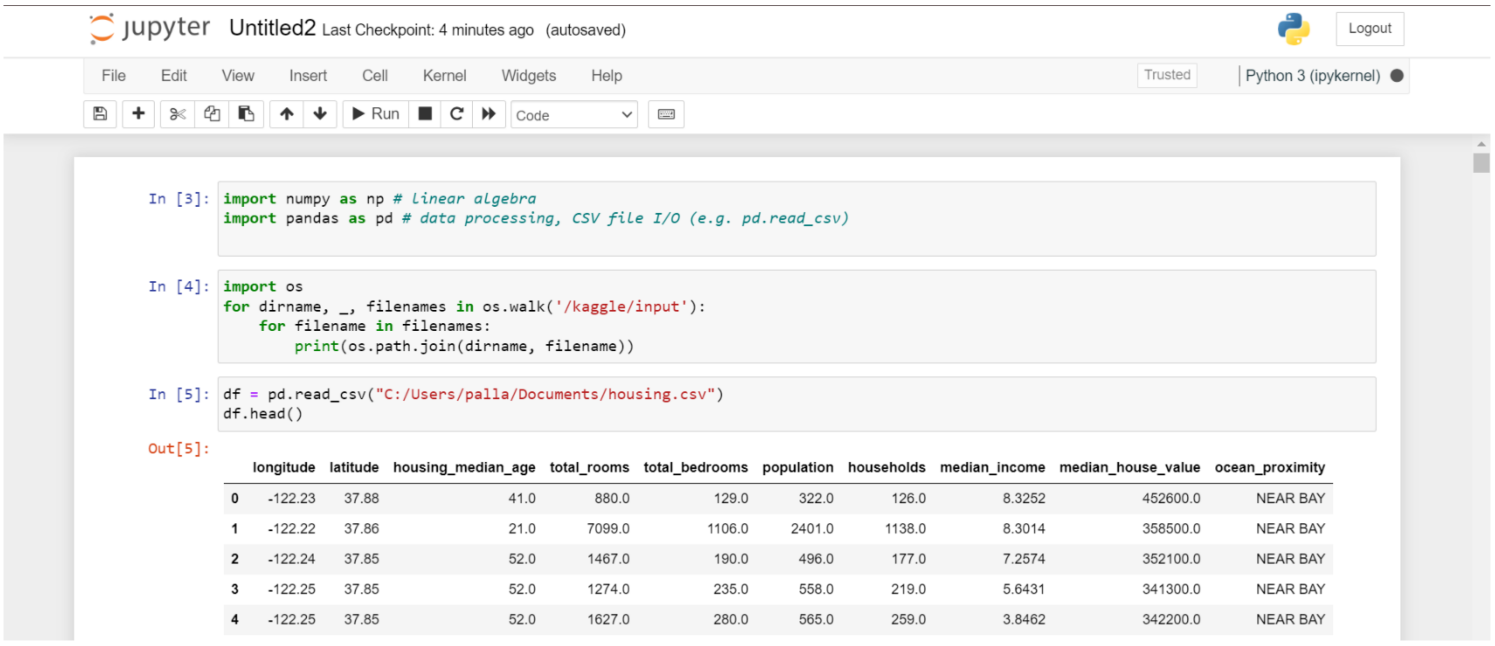
- Vanila autoencoders
- Convolutional auto encoders
- Recurrent auto encoders
- Variational auto encoders
- Denoising auto encoders
- Adversial auto encoders

Google net architecture:

Google net:
It is used in deep learning model which is developed by researchers of google and itconsist of 22-layers and trained on the image net dataset. It can classifyobjects into1000 different categories.

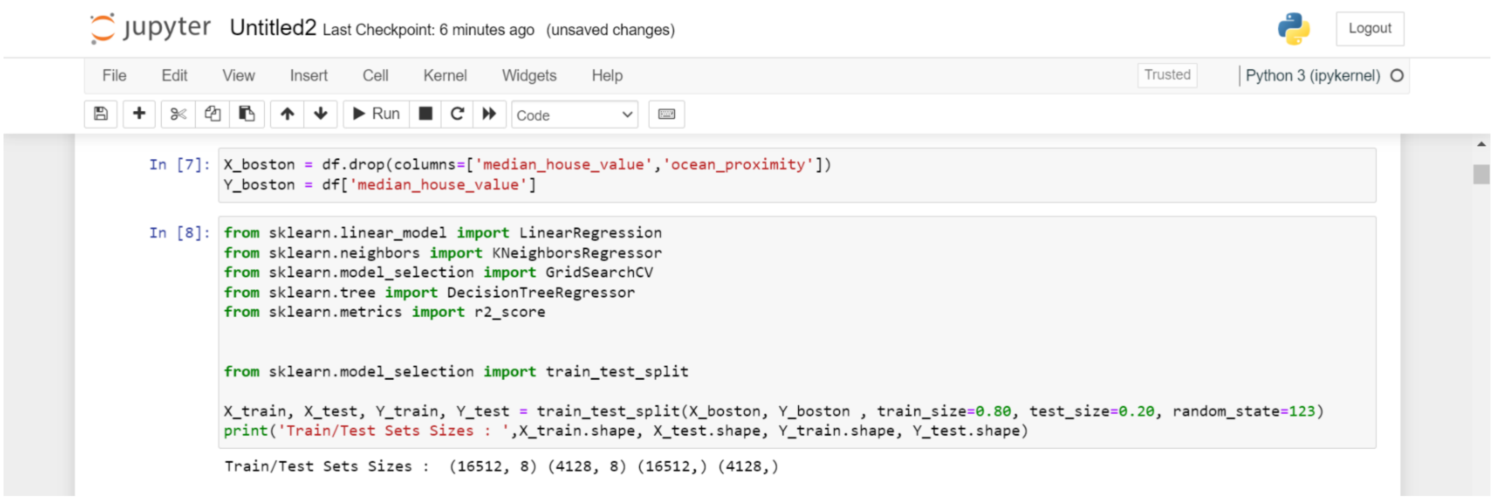
EDA FOR HOUSING PRICES PREDICTION :-

SOURCE CODE :-

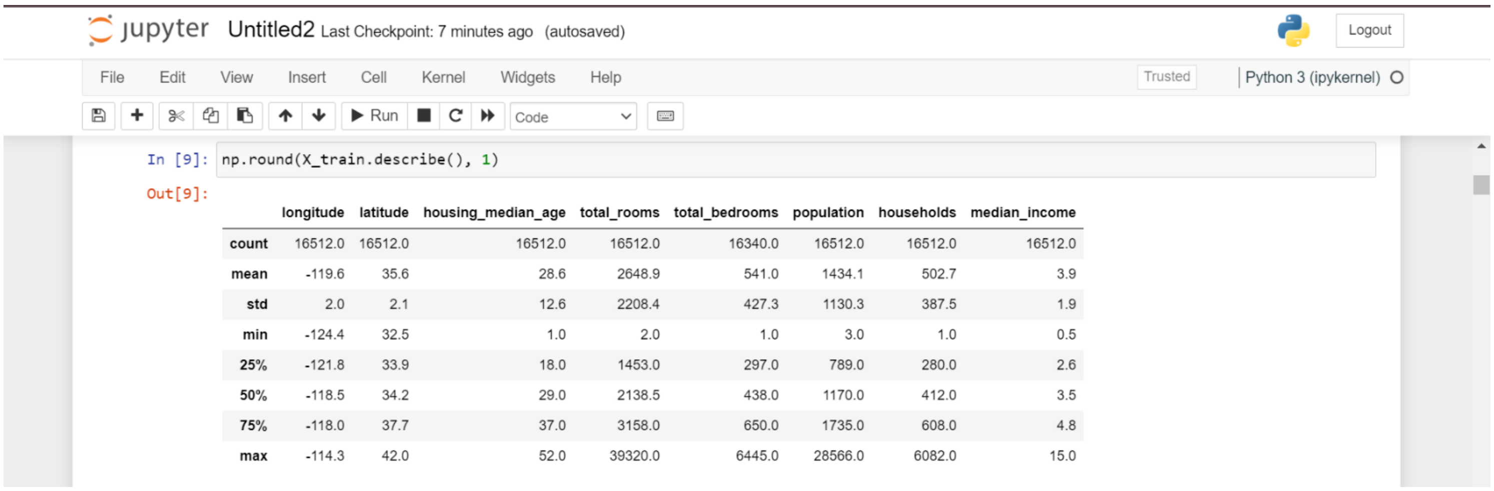


```
In [3]: import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

In [4]: import os
        for dirname, _, filenames in os.walk('/kaggle/input'):
            for filename in filenames:
                print(os.path.join(dirname, filename))

In [5]: df = pd.read_csv("C:/Users/palla/Documents/housing.csv")
        df.head()

Out[5]:
```

	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

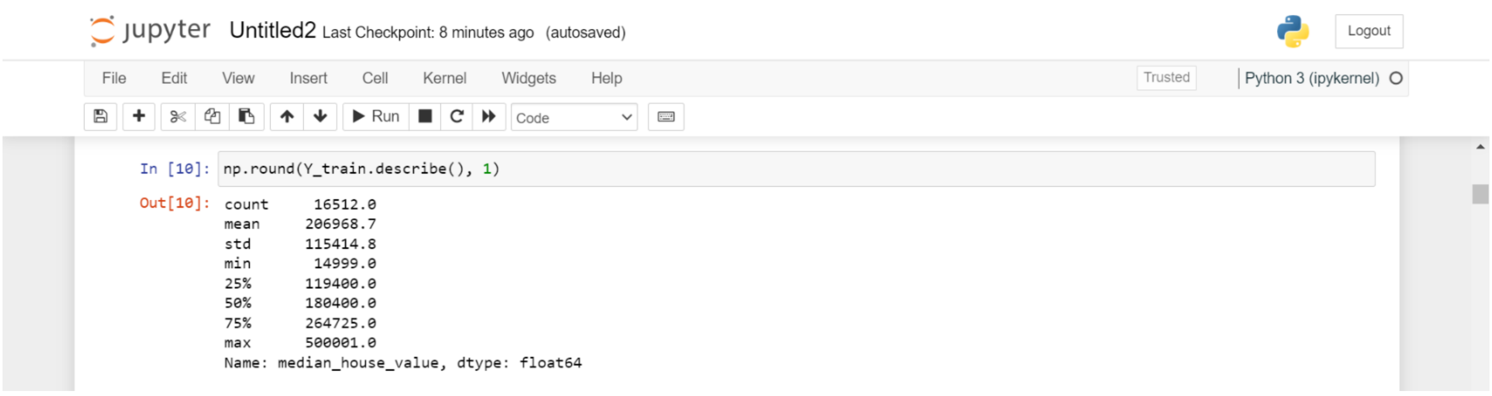


```
In [7]: X_boston = df.drop(columns=['median_house_value','ocean_proximity'])
        Y_boston = df['median_house_value']

In [8]: from sklearn.linear_model import LinearRegression
        from sklearn.neighbors import KNeighborsRegressor
        from sklearn.model_selection import GridSearchCV
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.metrics import r2_score


        from sklearn.model_selection import train_test_split

        X_train, X_test, Y_train, Y_test = train_test_split(X_boston, Y_boston , train_size=0.80, test_size=0.20, random_state=123)
        print('Train/Test Sets Sizes : ',X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)

        Train/Test Sets Sizes :  (16512, 8) (4128, 8) (16512,) (4128,)
```

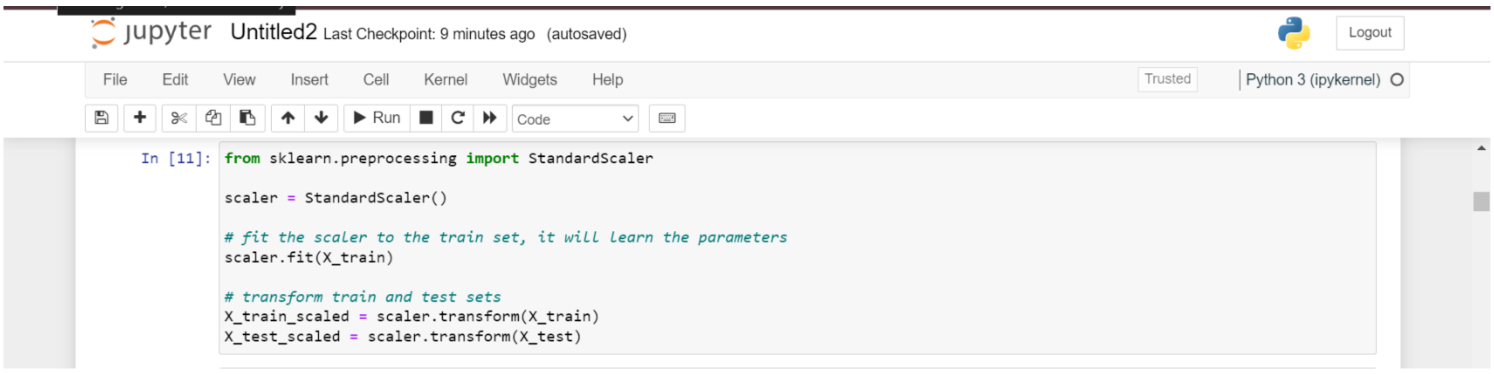

jupyter Untitled2 Last Checkpoint: 7 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [9]: np.round(X_train.describe(), 1)
Out[9]:
```

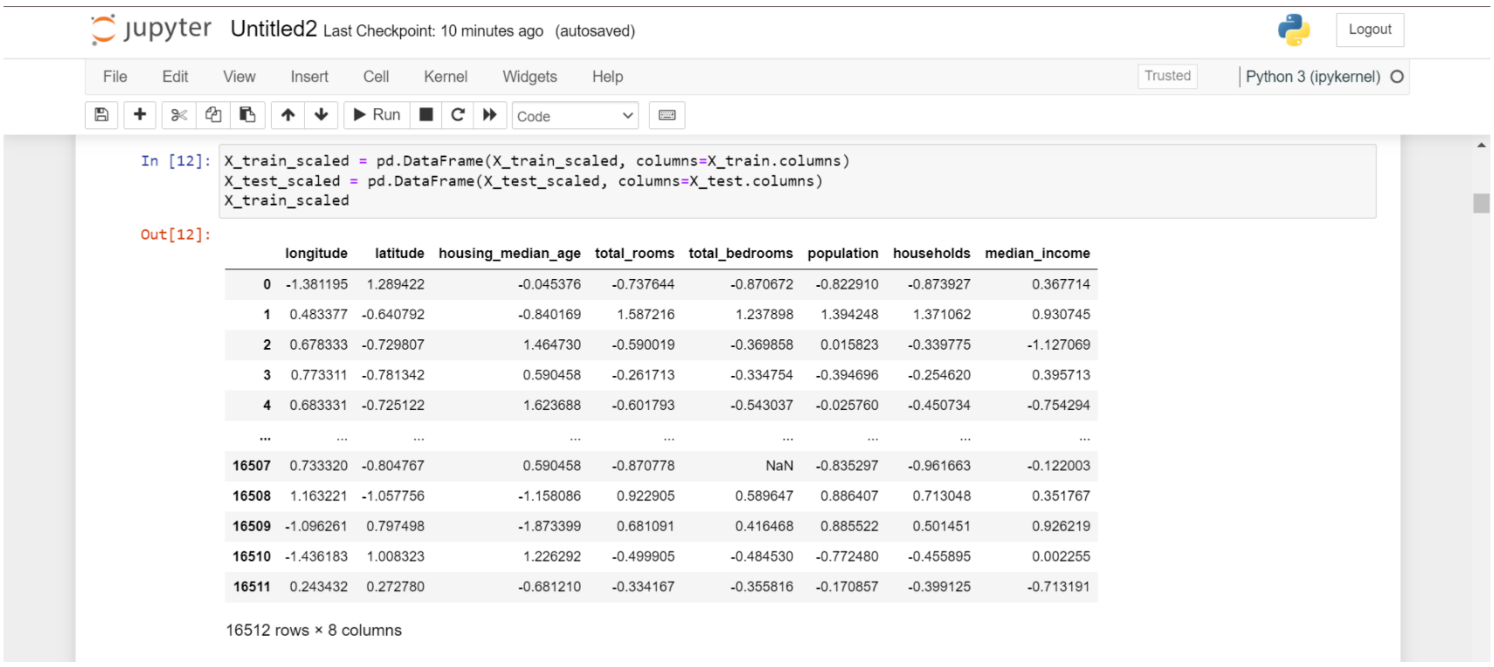
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
count	16512.0	16512.0	16512.0	16512.0	16340.0	16512.0	16512.0	16512.0
mean	-119.6	35.6	28.6	2648.9	541.0	1434.1	502.7	3.9
std	2.0	2.1	12.6	2208.4	427.3	1130.3	387.5	1.9
min	-124.4	32.5	1.0	2.0	1.0	3.0	1.0	0.5
25%	-121.8	33.9	18.0	1453.0	297.0	789.0	280.0	2.6
50%	-118.5	34.2	29.0	2138.5	438.0	1170.0	412.0	3.5
75%	-118.0	37.7	37.0	3158.0	650.0	1735.0	608.0	4.8
max	-114.3	42.0	52.0	39320.0	6445.0	28566.0	6082.0	15.0


jupyter Untitled2 Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [10]: np.round(Y_train.describe(), 1)
Out[10]: count     16512.0
         mean     206968.7
         std      115414.8
         min       14999.0
         25%      119400.0
         50%      180400.0
         75%      264725.0
         max      500001.0
         Name: median_house_value, dtype: float64
```


jupyter Untitled2 Last Checkpoint: 9 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [11]: from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()

         # fit the scaler to the train set, it will learn the parameters
         scaler.fit(X_train)

         # transform train and test sets
         X_train_scaled = scaler.transform(X_train)
         X_test_scaled = scaler.transform(X_test)
```

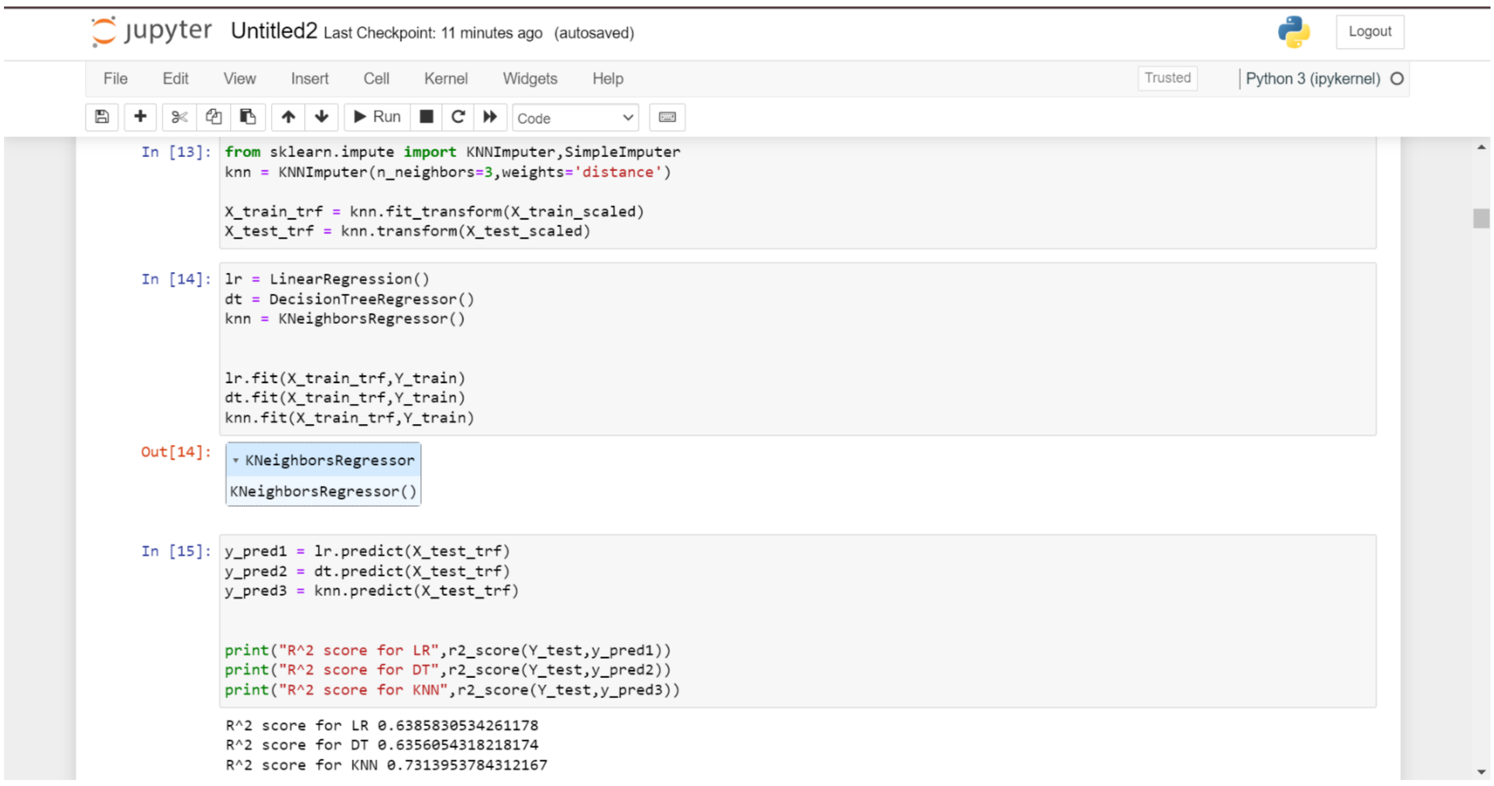

jupyter Untitled2 Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

In [12]:
```
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=X_test.columns)
X_train_scaled
```

Out[12]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
0	-1.381195	1.289422	-0.045376	-0.737644	-0.870672	-0.822910	-0.873927	0.367714
1	0.483377	-0.640792	-0.840169	1.587216	1.237898	1.394248	1.371062	0.930745
2	0.678333	-0.729807	1.464730	-0.590019	-0.369858	0.015823	-0.339775	-1.127069
3	0.773311	-0.781342	0.590458	-0.261713	-0.334754	-0.394696	-0.254620	0.395713
4	0.683331	-0.725122	1.623688	-0.601793	-0.543037	-0.025760	-0.450734	-0.754294
...	...	...	...	...	...	...	...	...
16507	0.733320	-0.804767	0.590458	-0.870778	NaN	-0.835297	-0.961663	-0.122003
16508	1.163221	-1.057756	-1.158086	0.922905	0.589647	0.886407	0.713048	0.351767
16509	-1.096261	0.797498	-1.873399	0.681091	0.416468	0.885522	0.501451	0.926219
16510	-1.436183	1.008323	1.226292	-0.499905	-0.484530	-0.772480	-0.455895	0.002255
16511	0.243432	0.272780	-0.681210	-0.334167	-0.355816	-0.170857	-0.399125	-0.713191

16512 rows × 8 columns


jupyter Untitled2 Last Checkpoint: 11 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

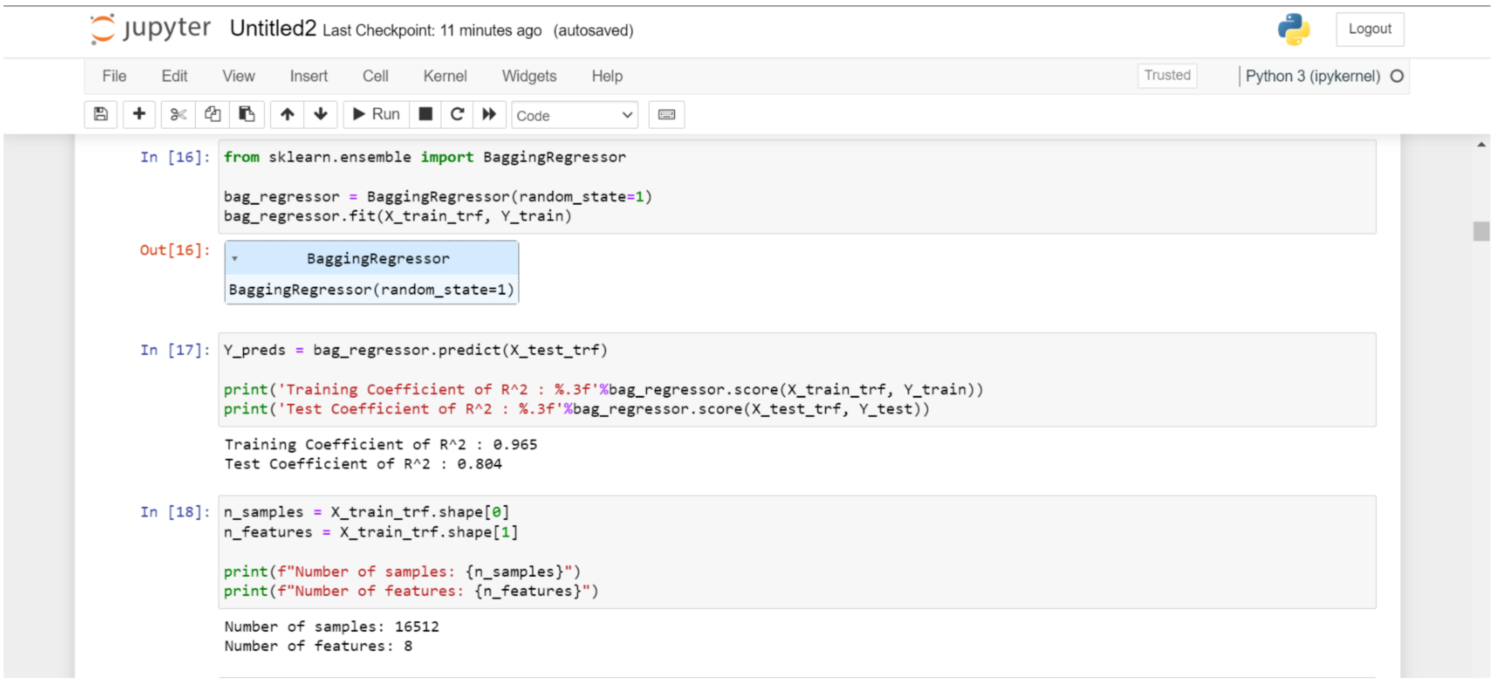
In [13]:
```
from sklearn.impute import KNNImputer,SimpleImputer
knn = KNNImputer(n_neighbors=3,weights='distance')

X_train_trf = knn.fit_transform(X_train_scaled)
X_test_trf = knn.transform(X_test_scaled)
```

In [14]:
```
lr = LinearRegression()
dt = DecisionTreeRegressor()
knn = KNeighborsRegressor()


lr.fit(X_train_trf,Y_train)
dt.fit(X_train_trf,Y_train)
knn.fit(X_train_trf,Y_train)
```

Out[14]:
```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

In [15]:
```
y_pred1 = lr.predict(X_test_trf)
y_pred2 = dt.predict(X_test_trf)
y_pred3 = knn.predict(X_test_trf)


print("R^2 score for LR",r2_score(Y_test,y_pred1))
print("R^2 score for DT",r2_score(Y_test,y_pred2))
print("R^2 score for KNN",r2_score(Y_test,y_pred3))
```

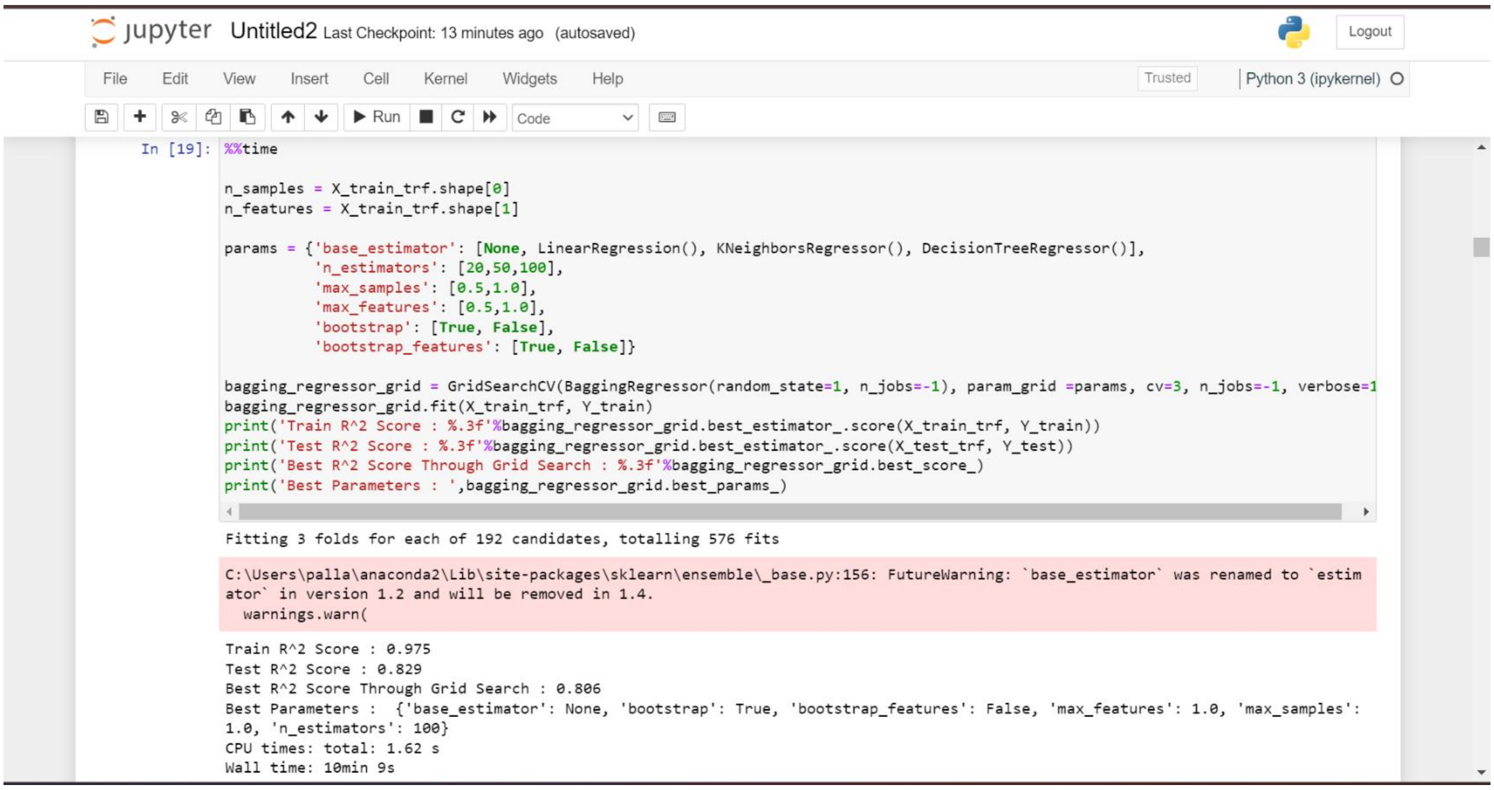
```
R^2 score for LR 0.6385830534261178
R^2 score for DT 0.6356054318218174
R^2 score for KNN 0.7313953784312167
```


jupyter Untitled2 Last Checkpoint: 11 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

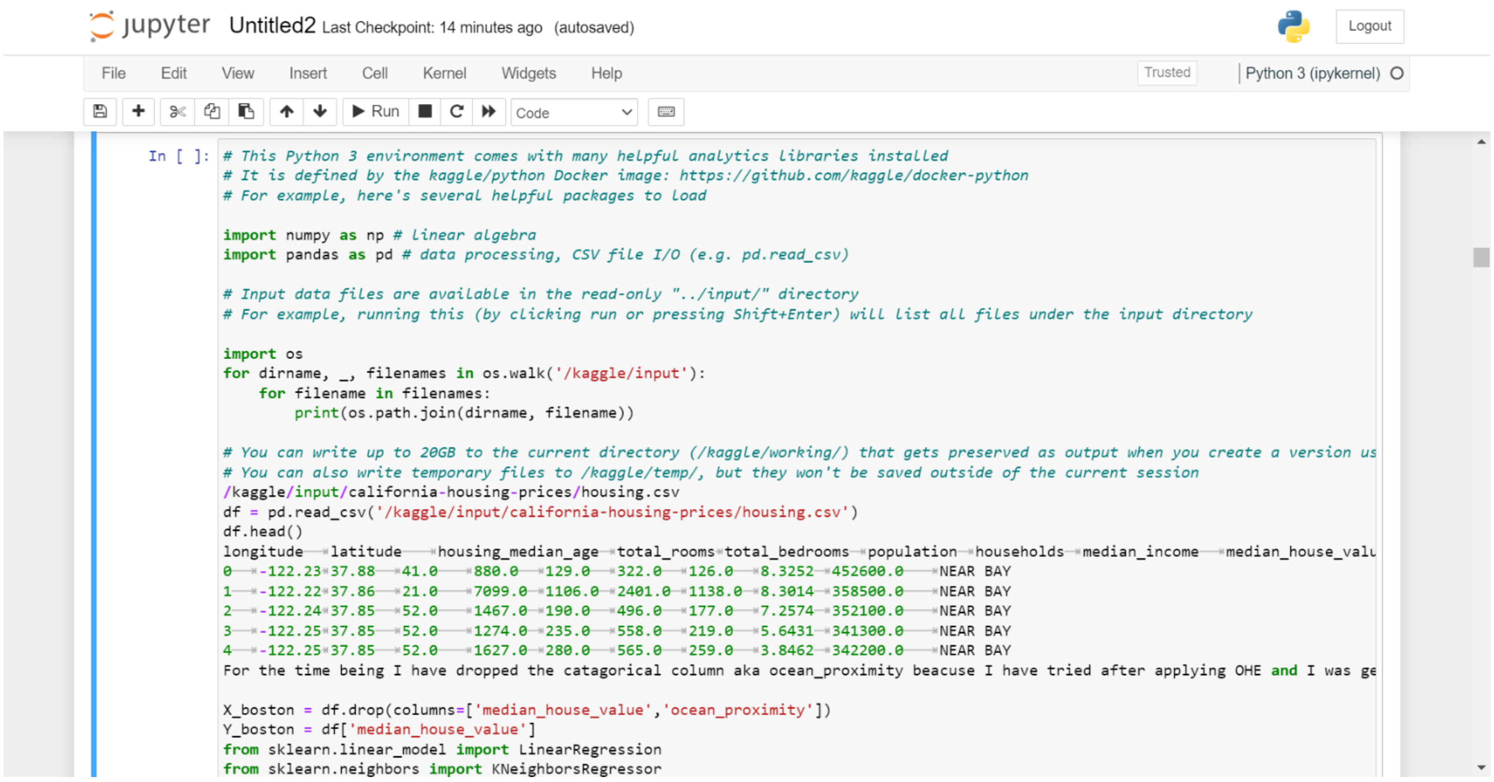
```
In [16]: from sklearn.ensemble import BaggingRegressor

         bag_regressor = BaggingRegressor(random_state=1)
         bag_regressor.fit(X_train_trf, Y_train)

Out[16]:         BaggingRegressor
         BaggingRegressor(random_state=1)
```

```
In [17]: Y_preds = bag_regressor.predict(X_test_trf)

         print('Training Coefficient of R^2 : %.3f'%bag_regressor.score(X_train_trf, Y_train))
         print('Test Coefficient of R^2 : %.3f'%bag_regressor.score(X_test_trf, Y_test))

         Training Coefficient of R^2 : 0.965
         Test Coefficient of R^2 : 0.804
```

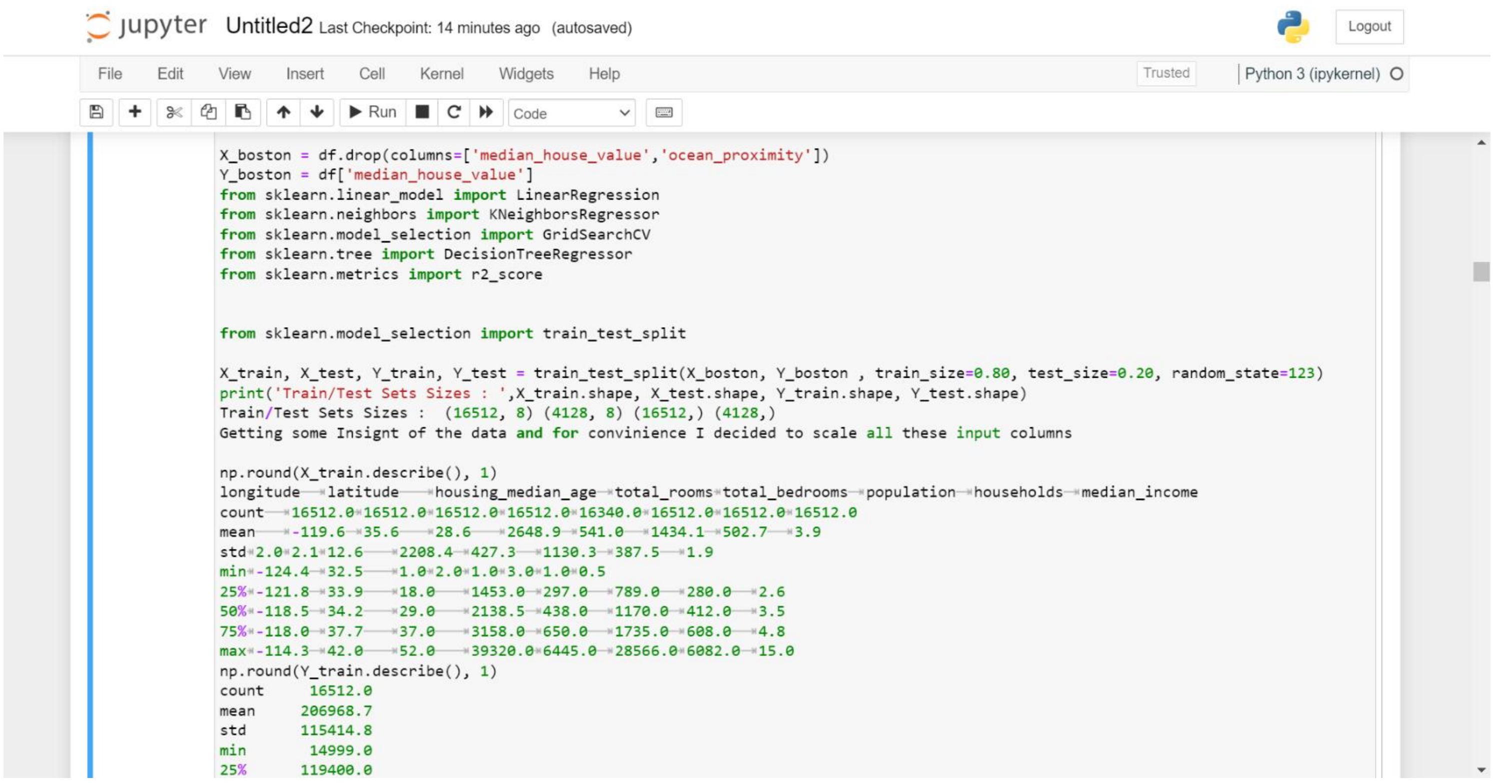
```
In [18]: n_samples = X_train_trf.shape[0]
         n_features = X_train_trf.shape[1]

         print(f"Number of samples: {n_samples}")
         print(f"Number of features: {n_features}")

         Number of samples: 16512
         Number of features: 8
```


jupyter Untitled2 Last Checkpoint: 13 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

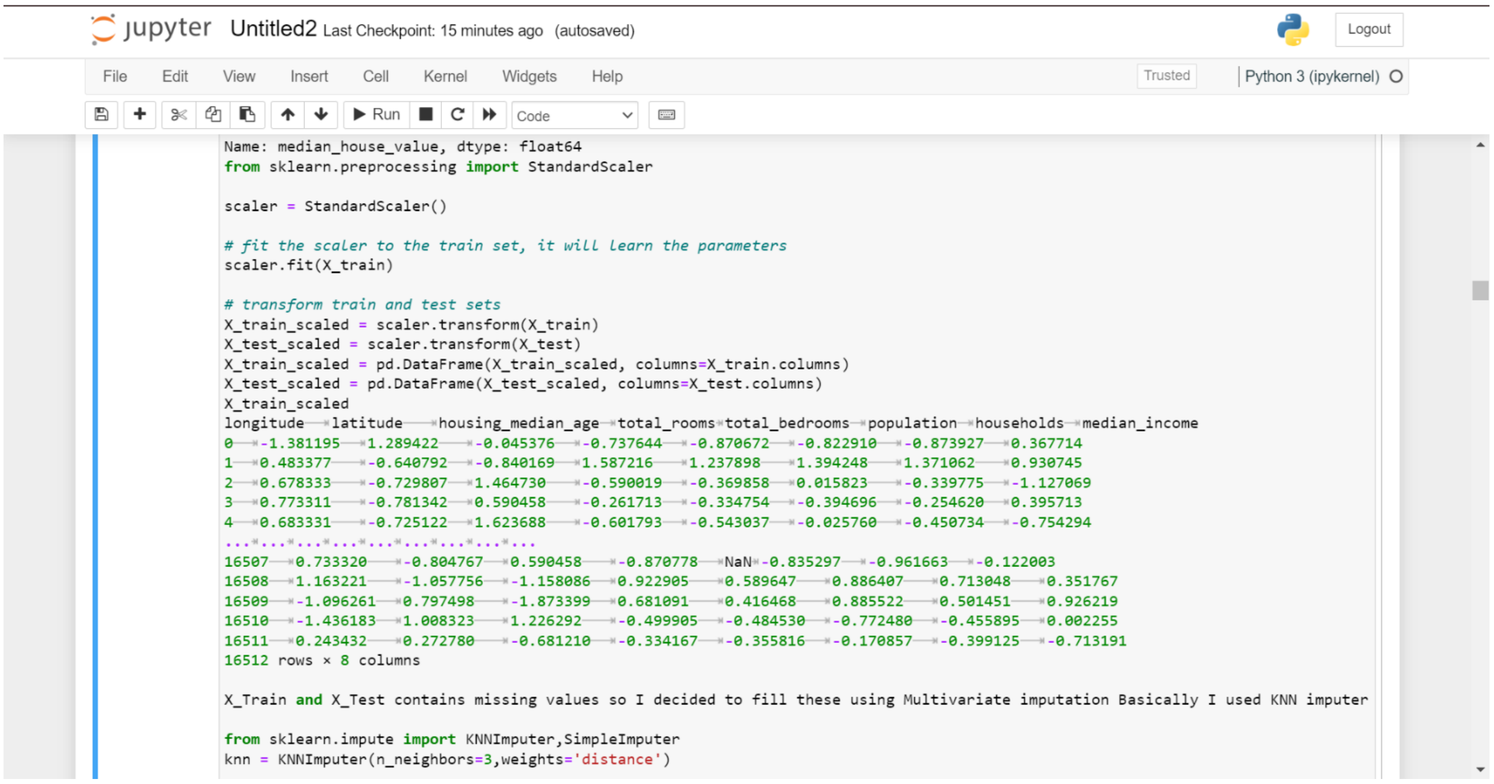
```
In [19]: %%time

         n_samples = X_train_trf.shape[0]
         n_features = X_train_trf.shape[1]

         params = {'base_estimator': [None, LinearRegression(), KNeighborsRegressor(), DecisionTreeRegressor()],
                   'n_estimators': [20,50,100],
                   'max_samples': [0.5,1.0],
                   'max_features': [0.5,1.0],
                   'bootstrap': [True, False],
                   'bootstrap_features': [True, False]}

         bagging_regressor_grid = GridSearchCV(BaggingRegressor(random_state=1, n_jobs=-1), param_grid =params, cv=3, n_jobs=-1, verbose=1
         bagging_regressor_grid.fit(X_train_trf, Y_train)
         print('Train R^2 Score : %.3f'%bagging_regressor_grid.best_estimator_.score(X_train_trf, Y_train))
         print('Test R^2 Score : %.3f'%bagging_regressor_grid.best_estimator_.score(X_test_trf, Y_test))
         print('Best R^2 Score Through Grid Search : %.3f'%bagging_regressor_grid.best_score_)
         print('Best Parameters : ',bagging_regressor_grid.best_params_)

         Fitting 3 folds for each of 192 candidates, totalling 576 fits

         C:\Users\palla\anaconda2\Lib\site-packages\sklearn\ensemble\_base.py:156: FutureWarning: `base_estimator` was renamed to `estim
         ator` in version 1.2 and will be removed in 1.4.
           warnings.warn(

         Train R^2 Score : 0.975
         Test R^2 Score : 0.829
         Best R^2 Score Through Grid Search : 0.806
         Best Parameters :  {'base_estimator': None, 'bootstrap': True, 'bootstrap_features': False, 'max_features': 1.0, 'max_samples':
         1.0, 'n_estimators': 100}
         CPU times: total: 1.62 s
         Wall time: 10min 9s
```



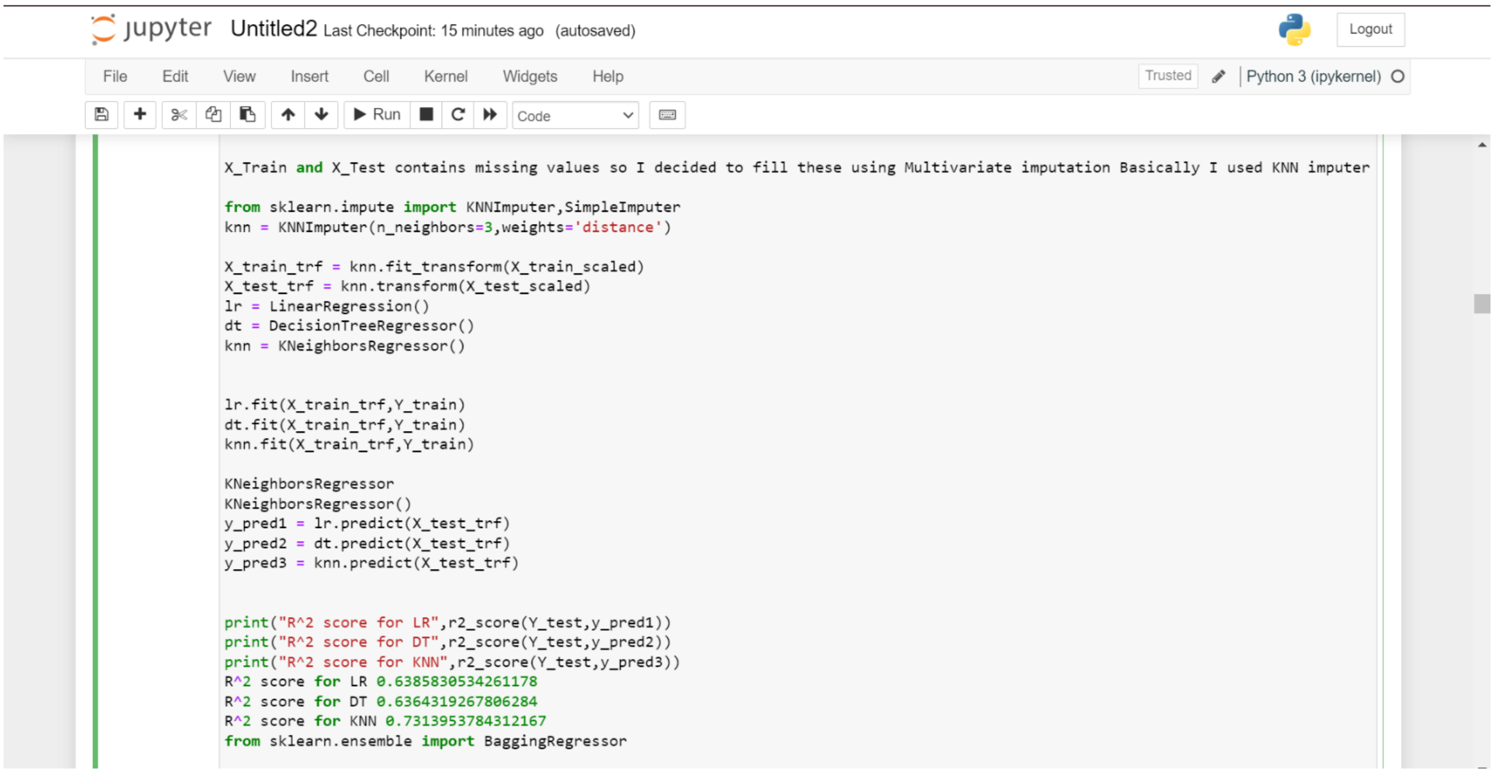
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version us
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
/kaggle/input/california-housing-prices/housing.csv
df = pd.read_csv('/kaggle/input/california-housing-prices/housing.csv')
df.head()
longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_valu
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
For the time being I have dropped the catagorical column aka ocean_proximity beacuse I have tried after applying OHE and I was ge

X_boston = df.drop(columns=['median_house_value','ocean_proximity'])
Y_boston = df['median_house_value']
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
```



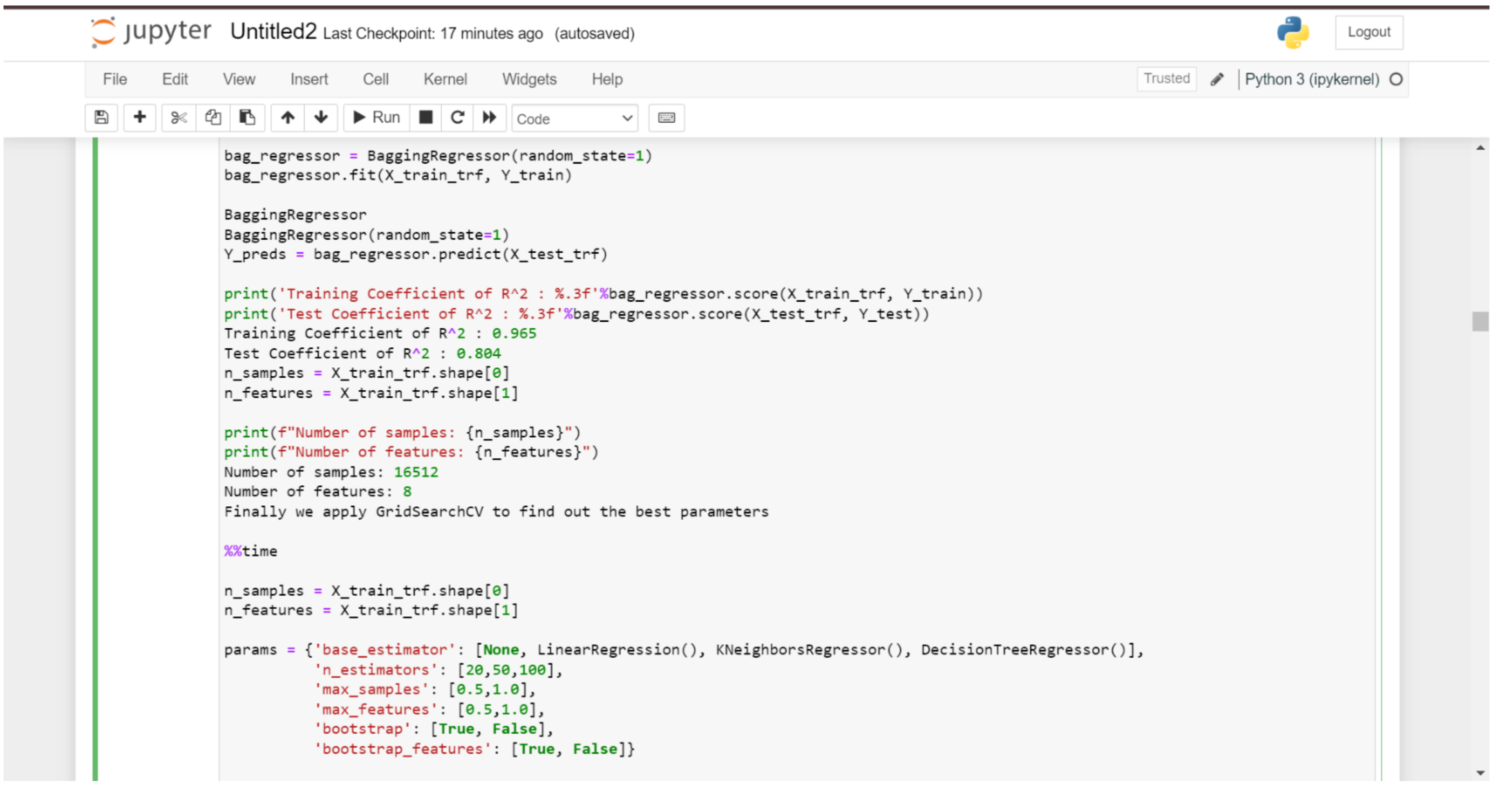
```
X_boston = df.drop(columns=['median_house_value','ocean_proximity'])
Y_boston = df['median_house_value']
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score


from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X_boston, Y_boston , train_size=0.80, test_size=0.20, random_state=123)
print('Train/Test Sets Sizes : ',X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
Train/Test Sets Sizes :  (16512, 8) (4128, 8) (16512,) (4128,)
Getting some Insight of the data and for convinience I decided to scale all these input columns

np.round(X_train.describe(), 1)
longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
count	16512.0	16512.0	16512.0	16512.0	16340.0	16512.0	16512.0	16512.0
mean	-119.6	35.6	28.6	2648.9	541.0	1434.1	502.7	3.9
std	2.0	2.1	12.6	2208.4	427.3	1130.3	387.5	1.9
min	-124.4	32.5	1.0	2.0	1.0	3.0	1.0	0.5
25%	-121.8	33.9	18.0	1453.0	297.0	789.0	280.0	2.6
50%	-118.5	34.2	29.0	2138.5	438.0	1170.0	412.0	3.5
75%	-118.0	37.7	37.0	3158.0	650.0	1735.0	608.0	4.8
max	-114.3	42.0	52.0	39320.0	6445.0	28566.0	6082.0	15.0
np.round(Y_train.describe(), 1)
count     16512.0
mean     206968.7
std      115414.8
min       14999.0
25%      119400.0
```


jupyter Untitled2 Last Checkpoint: 15 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

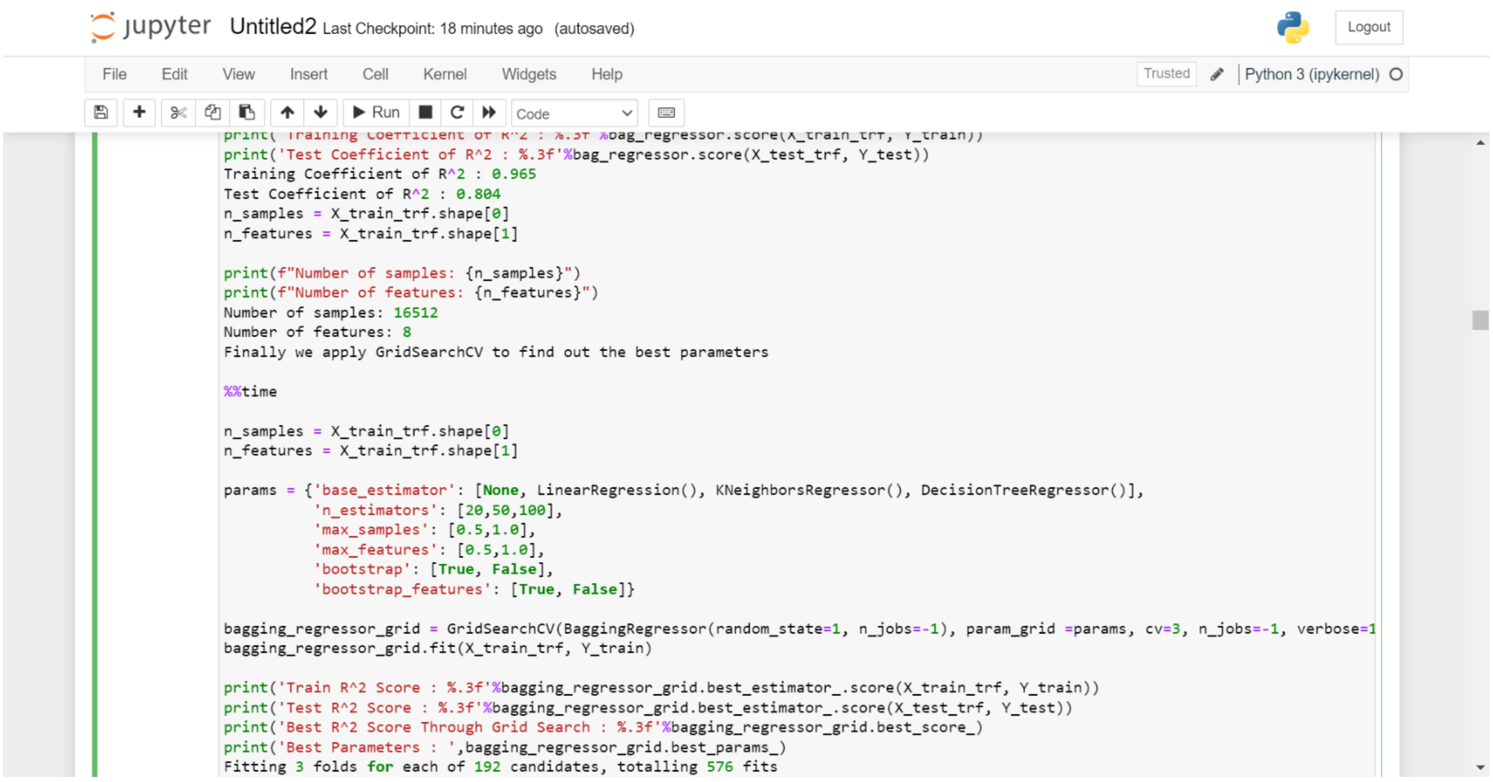
```
Name: median_house_value, dtype: float64
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# fit the scaler to the train set, it will learn the parameters
scaler.fit(X_train)

# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=X_test.columns)
X_train_scaled
longitude   latitude    housing_median_age  total_rooms total_bedrooms  population  households  median_income
0   -1.381195   1.289422    -0.045376   -0.737644   -0.870672   -0.822910   -0.873927   0.367714
1   0.483377    -0.640792   -0.840169   1.587216    1.237898    1.394248    1.371062    0.930745
2   0.678333    -0.729807   1.464730    -0.590019   -0.369858   0.015823    -0.339775   -1.127069
3   0.773311    -0.781342   0.590458    -0.261713   -0.334754   -0.394696   -0.254620   0.395713
4   0.683331    -0.725122   1.623688    -0.601793   -0.543037   -0.025760   -0.450734   -0.754294
... ... ... ... ... ... ... ... ...
16507   0.733320    -0.804767   0.590458    -0.870778   NaN -0.835297   -0.961663   -0.122003
16508   1.163221    -1.057756   -1.158086   0.922905    0.589647    0.886407    0.713048    0.351767
16509   -1.096261   0.797498    -1.873399   0.681091    0.416468    0.885522    0.501451    0.926219
16510   -1.436183   1.008323    1.226292    -0.499905   -0.484530   -0.772480   -0.455895   0.002255
16511   0.243432    0.272780    -0.681210   -0.334167   -0.355816   -0.170857   -0.399125   -0.713191
16512 rows × 8 columns

X_Train and X_Test contains missing values so I decided to fill these using Multivariate imputation Basically I used KNN imputer

from sklearn.impute import KNNImputer,SimpleImputer
knn = KNNImputer(n_neighbors=3,weights='distance')
```


jupyter Untitled2 Last Checkpoint: 15 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

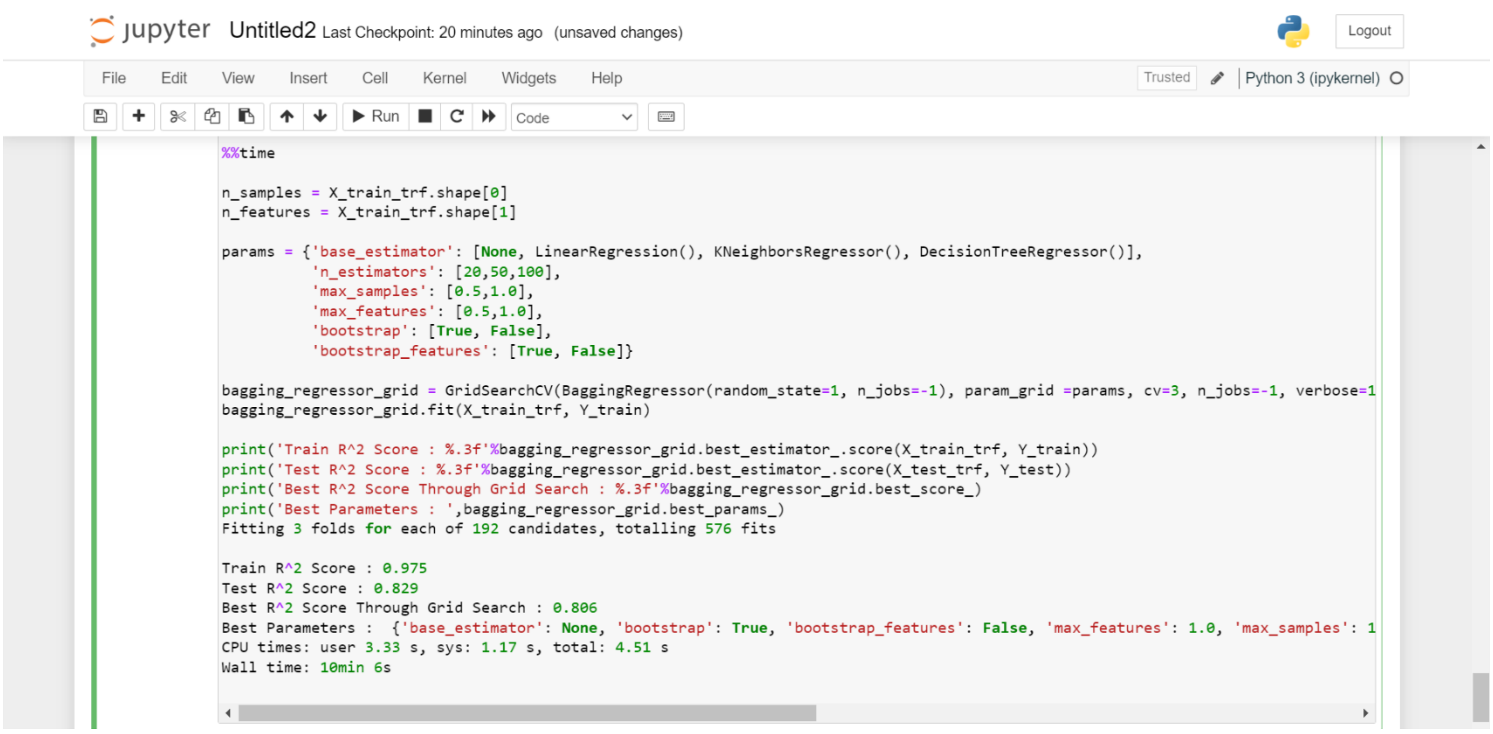
```
X_Train and X_Test contains missing values so I decided to fill these using Multivariate imputation Basically I used KNN imputer

from sklearn.impute import KNNImputer,SimpleImputer
knn = KNNImputer(n_neighbors=3,weights='distance')

X_train_trf = knn.fit_transform(X_train_scaled)
X_test_trf = knn.transform(X_test_scaled)
lr = LinearRegression()
dt = DecisionTreeRegressor()
knn = KNeighborsRegressor()


lr.fit(X_train_trf,Y_train)
dt.fit(X_train_trf,Y_train)
knn.fit(X_train_trf,Y_train)

KNeighborsRegressor
KNeighborsRegressor()
y_pred1 = lr.predict(X_test_trf)
y_pred2 = dt.predict(X_test_trf)
y_pred3 = knn.predict(X_test_trf)


print("R^2 score for LR",r2_score(Y_test,y_pred1))
print("R^2 score for DT",r2_score(Y_test,y_pred2))
print("R^2 score for KNN",r2_score(Y_test,y_pred3))
R^2 score for LR 0.6385830534261178
R^2 score for DT 0.6364319267806284
R^2 score for KNN 0.7313953784312167
from sklearn.ensemble import BaggingRegressor
```



```
bag_regressor = BaggingRegressor(random_state=1)
bag_regressor.fit(X_train_trf, Y_train)

BaggingRegressor
BaggingRegressor(random_state=1)
Y_preds = bag_regressor.predict(X_test_trf)

print('Training Coefficient of R^2 : %.3f'%bag_regressor.score(X_train_trf, Y_train))
print('Test Coefficient of R^2 : %.3f'%bag_regressor.score(X_test_trf, Y_test))
Training Coefficient of R^2 : 0.965
Test Coefficient of R^2 : 0.804
n_samples = X_train_trf.shape[0]
n_features = X_train_trf.shape[1]

print(f"Number of samples: {n_samples}")
print(f"Number of features: {n_features}")
Number of samples: 16512
Number of features: 8
Finally we apply GridSearchCV to find out the best parameters

%%time

n_samples = X_train_trf.shape[0]
n_features = X_train_trf.shape[1]

params = {'base_estimator': [None, LinearRegression(), KNeighborsRegressor(), DecisionTreeRegressor()],
          'n_estimators': [20,50,100],
          'max_samples': [0.5,1.0],
          'max_features': [0.5,1.0],
          'bootstrap': [True, False],
          'bootstrap_features': [True, False]}
```

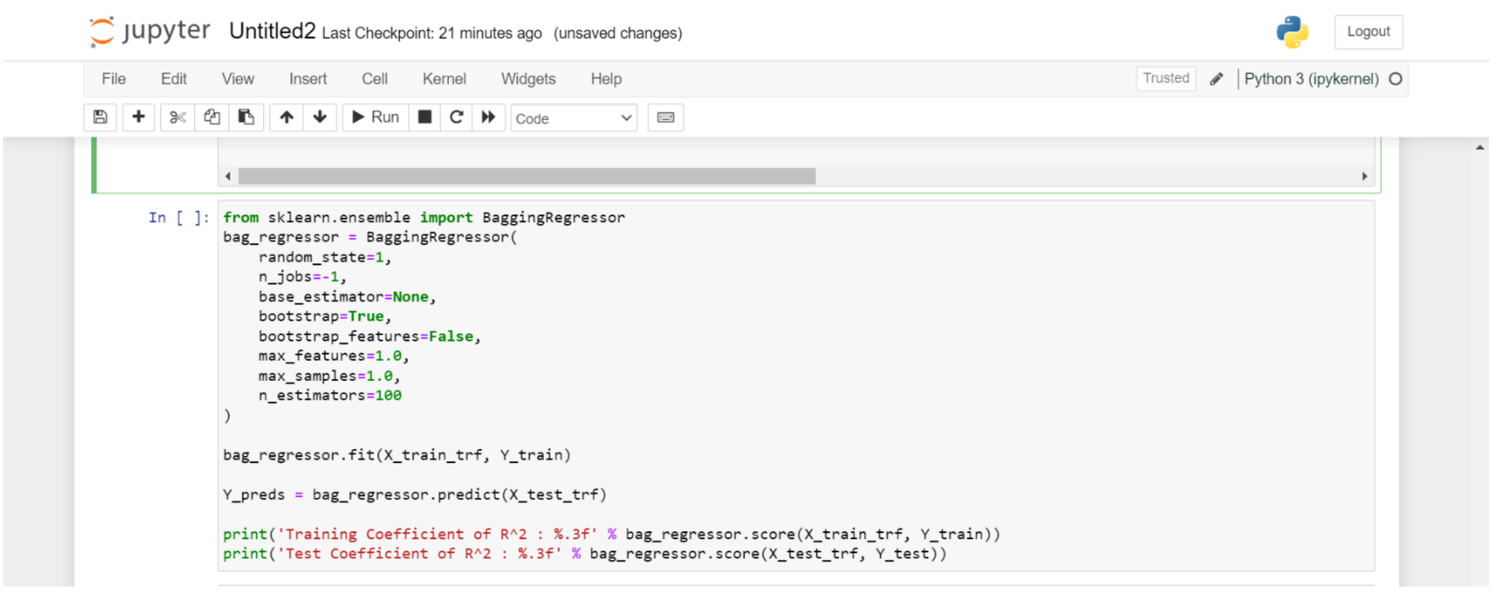


```
print('Training Coefficient of R^2 : %.3f'%bag_regressor.score(X_train_trf, Y_train))
print('Test Coefficient of R^2 : %.3f'%bag_regressor.score(X_test_trf, Y_test))
Training Coefficient of R^2 : 0.965
Test Coefficient of R^2 : 0.804
n_samples = X_train_trf.shape[0]
n_features = X_train_trf.shape[1]

print(f"Number of samples: {n_samples}")
print(f"Number of features: {n_features}")
Number of samples: 16512
Number of features: 8
Finally we apply GridSearchCV to find out the best parameters

%%time

n_samples = X_train_trf.shape[0]
n_features = X_train_trf.shape[1]

params = {'base_estimator': [None, LinearRegression(), KNeighborsRegressor(), DecisionTreeRegressor()],
          'n_estimators': [20,50,100],
          'max_samples': [0.5,1.0],
          'max_features': [0.5,1.0],
          'bootstrap': [True, False],
          'bootstrap_features': [True, False]}

bagging_regressor_grid = GridSearchCV(BaggingRegressor(random_state=1, n_jobs=-1), param_grid =params, cv=3, n_jobs=-1, verbose=1
bagging_regressor_grid.fit(X_train_trf, Y_train)

print('Train R^2 Score : %.3f'%bagging_regressor_grid.best_estimator_.score(X_train_trf, Y_train))
print('Test R^2 Score : %.3f'%bagging_regressor_grid.best_estimator_.score(X_test_trf, Y_test))
print('Best R^2 Score Through Grid Search : %.3f'%bagging_regressor_grid.best_score_)
print('Best Parameters : ',bagging_regressor_grid.best_params_)
Fitting 3 folds for each of 192 candidates, totalling 576 fits
```



jupyter Untitled2 Last Checkpoint: 20 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
%%time

n_samples = X_train_trf.shape[0]
n_features = X_train_trf.shape[1]

params = {'base_estimator': [None, LinearRegression(), KNeighborsRegressor(), DecisionTreeRegressor()],
          'n_estimators': [20,50,100],
          'max_samples': [0.5,1.0],
          'max_features': [0.5,1.0],
          'bootstrap': [True, False],
          'bootstrap_features': [True, False]}

bagging_regressor_grid = GridSearchCV(BaggingRegressor(random_state=1, n_jobs=-1), param_grid =params, cv=3, n_jobs=-1, verbose=1
bagging_regressor_grid.fit(X_train_trf, Y_train)

print('Train R^2 Score : %.3f'%bagging_regressor_grid.best_estimator_.score(X_train_trf, Y_train))
print('Test R^2 Score : %.3f'%bagging_regressor_grid.best_estimator_.score(X_test_trf, Y_test))
print('Best R^2 Score Through Grid Search : %.3f'%bagging_regressor_grid.best_score_)
print('Best Parameters : ',bagging_regressor_grid.best_params_)
Fitting 3 folds for each of 192 candidates, totalling 576 fits

Train R^2 Score : 0.975
Test R^2 Score : 0.829
Best R^2 Score Through Grid Search : 0.806
Best Parameters :  {'base_estimator': None, 'bootstrap': True, 'bootstrap_features': False, 'max_features': 1.0, 'max_samples': 1
CPU times: user 3.33 s, sys: 1.17 s, total: 4.51 s
Wall time: 10min 6s
```



jupyter Untitled2 Last Checkpoint: 21 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [ ]: from sklearn.ensemble import BaggingRegressor
        bag_regressor = BaggingRegressor(
            random_state=1,
            n_jobs=-1,
            base_estimator=None,
            bootstrap=True,
            bootstrap_features=False,
            max_features=1.0,
            max_samples=1.0,
            n_estimators=100
        )

        bag_regressor.fit(X_train_trf, Y_train)

        Y_preds = bag_regressor.predict(X_test_trf)

        print('Training Coefficient of R^2 : %.3f' % bag_regressor.score(X_train_trf, Y_train))
        print('Test Coefficient of R^2 : %.3f' % bag_regressor.score(X_test_trf, Y_test))
```

CONCLUSION:

Conclusion for Exploratory Data Analysis in California Housing Prices :-

Exploratory Data Analysis (EDA) is an essential process for gaining insights into Housing prices data. Through EDA, we can uncover patterns, trends, and relationships that are crucial for informed decision-making in the retail sector.

In conclusion, the California housing prices dataset stands as a vital repository of information, offering profound insights into one of the most dynamic real estate markets in the United States. Through comprehensive Exploratory Data Analysis (EDA), we have unraveled a multitude of trends, patterns, and relationships that shape housing prices across the state.

From understanding the impact of economic factors such as income levels and employment rates to exploring spatial variations in property values and demographic shifts, the dataset has provided a nuanced perspective on the complex interplay of variables influencing California's housing market. EDA has enabled us to detect outliers, assess data quality, and identify key predictors, laying the groundwork for predictive modeling and deeper statistical analyses.

Moreover, the insights gained from EDA are not merely academic but hold practical significance for stakeholders ranging from policymakers and urban planners to real estate developers and investors. By leveraging these insights, stakeholders can make informed decisions, formulate effective policies, and navigate the challenges of housing affordability and market volatility in California.

Looking forward, continued exploration and refinement of the dataset through advanced analytical techniques promise to unveil further insights and opportunities for enhancing housing accessibility, sustainability, and economic resilience across the diverse landscapes of California. Ultimately, the California housing prices dataset stands as a testament to the power of data-driven insights in shaping a more equitable and prosperous future for residents and investors alike in the Golden State.