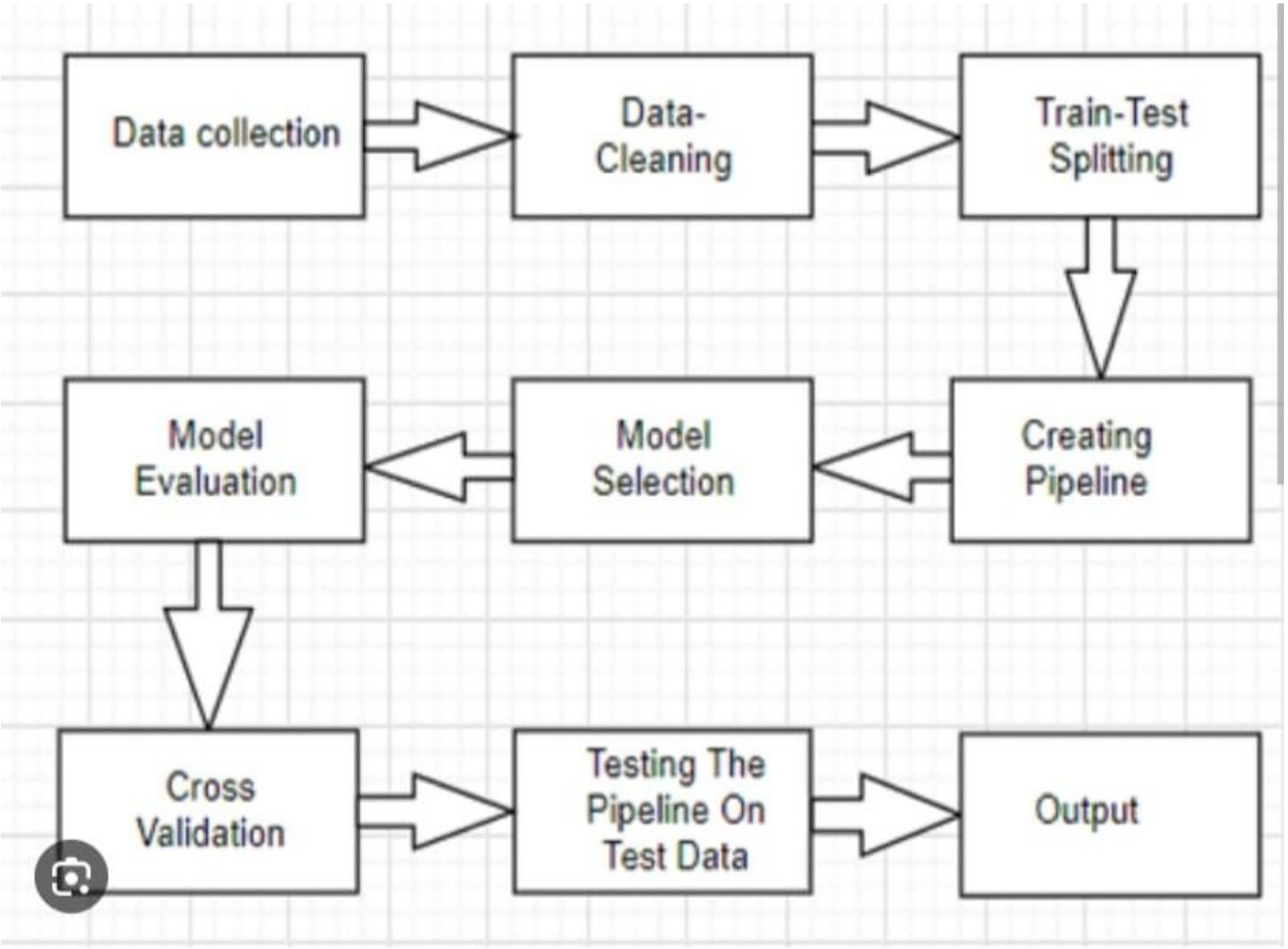
Development of predicting House prices using Machine learning


Data collection
Data-
Cleaning
Train-Test
Splitting
Creating
Pipeline
Model
Selection
Model
Evaluation
Cross
Validation
Testing The
Pipeline On
Test Data
Output

Certainly! To continue building a house price prediction model, you can use Python and popular libraries like scikit-learn and pandas for feature selection, model training, and evaluation. Here's a basic example with comments for each step:

pythonCopy code

Import necessary libraries import pandas as pd from sklearn.model_selection import train_test_split from sklearn.ensemble import RandomForestRegressor from sklearn.metrics import mean_squared_error

Load your dataset (replace 'data.csv' with your dataset file) data = pd.read_csv('data.csv') # Feature selection: Choose relevant features

In this example, we select 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', and 'yr_built' selected_features = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'yr_built'] X = data[selected_features] y = data['price']

Split the dataset 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)

Create and train the model (Random Forest Regressor in this case) model = RandomForestRegressor(n_estimators=100, random_state=42) model.fit(X_train, y_train)

Make predictions on the test set y_pred = model.predict(X_test)

Evaluate the model using Mean Squared Error (MSE) mse = mean_squared_error(y_test, y_pred) print(f"Mean Squared Error: {mse}")

You can further fine-tune your model and evaluate its performance using various metrics and techniques.

This is a basic example, and you can customize it according to your dataset and requirements. You may consider other regression models, hyperparameter tuning, feature engineering, and cross-validation for more robust predictions.

PREDICTING HOUSE PRICES USING MACHINE LEARNING

Introduction

Welcome to the presentation on *Unveiling the Future: Harnessing the Power of Machine Learning to Predict House Prices*. In this session, we will explore how **machine learning** algorithms can accurately predict house prices, enabling better decision-making in the real estate industry.

What is Machine Learning?

Machine Learning is a subset of **artificial intelligence** that enables computers to learn and make predictions without being explicitly programmed. It leverages **statistical models** and **algorithms** to analyze data and identify patterns, allowing us to make accurate predictions.

Importance of House Price Prediction

Accurate house price prediction is crucial for various stakeholders in the real estate industry, including **homebuyers**, **sellers**, and **investors**. It helps in making informed decisions, understanding market trends, and maximizing returns on investments.

TRAINING AND TESTING

To test the accuracy of the model we use a metric mean_absolute_percentage_error which we import from scikit library. The algorithms used for training and testing the model are as follows-

- KNN- First we import the KNeighborsRegressor() function from scikit library and then run the algorithm on training set and then on test set.
- We obtain the mean absolute percentage error of knn as 38.91504096125152 %

Spliting into training and testing data

```
In [28]: x = data.drop(columns=["longitude","latitude","median_house_value"])
         y = data["median_house_value"]

In [29]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test = train_test_split(x,y, random_state = 1)
```

KN-Neighbors

```
In [30]: from sklearn.neighbors import KNeighborsRegressor

In [31]: knn = KNeighborsRegressor()
         knn.fit(x_train, y_train)

Out[31]: KNeighborsRegressor()

In [32]: b = knn.predict(x_test)

In [33]: print("mean_absolute_percentage_error of KNN model:",mean_absolute_percentage_error(b,y_test)*100,"%")

         mean_absolute_percentage_error of KNN model: 38.91504096125152 %
```

- SVM- Similarly we import SVR() from scikit library.
- Then we run the svm algorithm on the training data and then the test data.
- The mean absolute percentage error of svm obtained is 32.246431895552135 %, which is slightly less than that of knn algorithm.

- ANN- We use tensor flow library to build an ANN here for the application.
- We also use the Keras library here which is an open source deep learning framework for python. Keras is one of the most powerful and easy to use python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, etc., for creating deep learning models.

- To build the different layers of ANN we use ‘Flatten’ and ‘Dense’ function of keras framework.
- Dense layer is the regular deeply connected neural network layer. It is most common and frequently used layer.
- Flatten layer is used to flatten the input.
- Then after building the layers of the model we need to compile the model so that the code is converted into machine readable code.
- We can view the summary of the model using model.summary() function.

Support Vector Machine

In [34]:
```
from sklearn.svm import SVR
```

In [35]:
```
svm = SVR(C = 2, kernel = "linear")
svm.fit(x_train, y_train)
```

Out[35]:
```
SVR(C=2, kernel='linear')
```

In [36]:
```
c = svm.predict(x_test)
```

In [37]:
```
print("mean_absolute_percentage_error of SVM model: ",mean_absolute_percentage_error(c,y_test)*100,"%")
```

```
mean_absolute_percentage_error of SVM model:  32.246431895552135 %
```

In [38]:
```
import tensorflow as tf
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=x.shape[1:]),
  tf.keras.layers.Dense(256, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dense(16, activation='relu'),
  tf.keras.layers.Dense(8, activation='relu'),
  tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam',
              loss='mean_squared_error')
model.summary()
```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 12)	0
dense (Dense)	(None, 256)	3328
dense_1 (Dense)	(None, 128)	32896
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 32)	2080
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 8)	136
dense_6 (Dense)	(None, 1)	9

Total params: 47,233
Trainable params: 47,233
Non-trainable params: 0

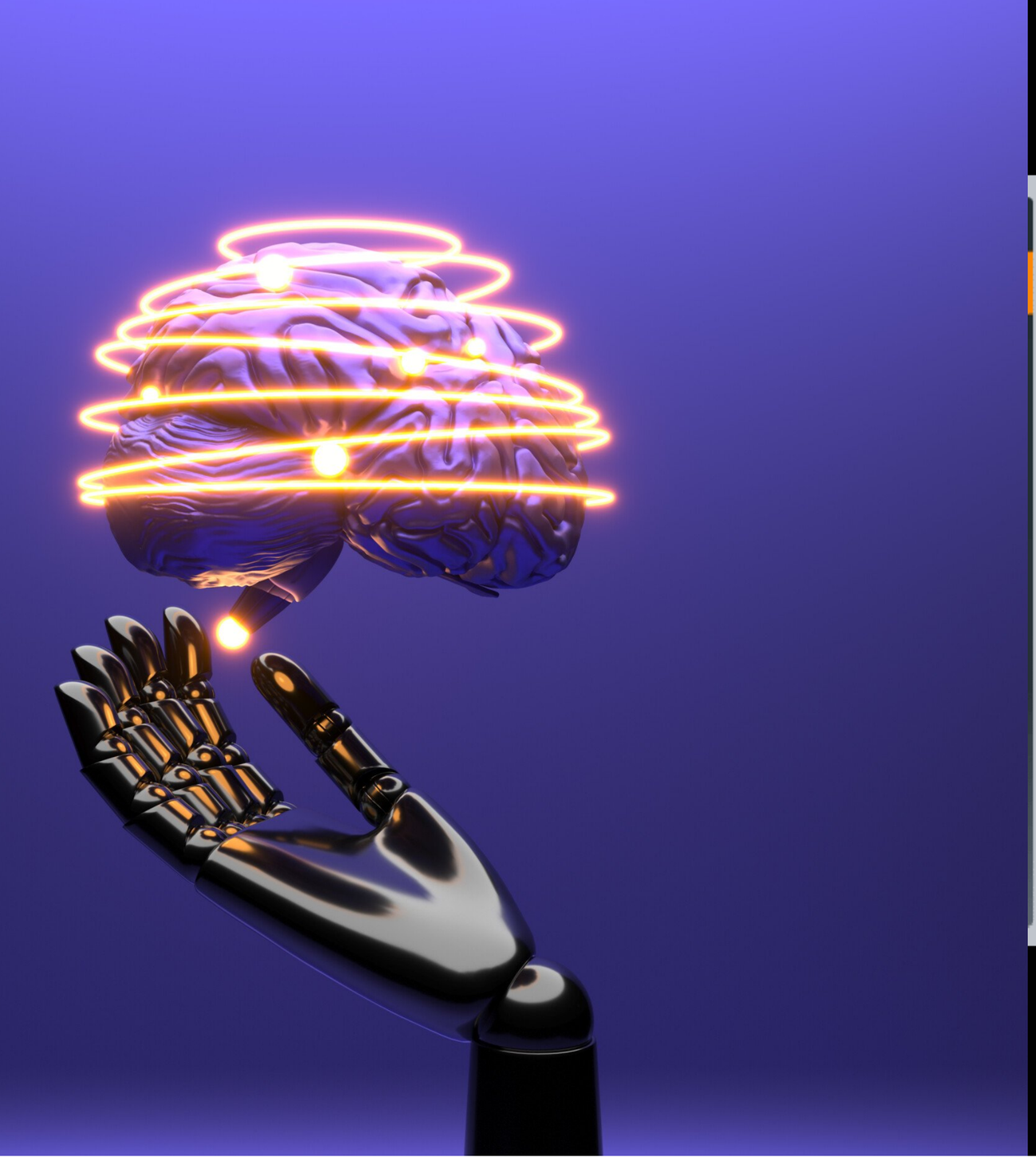
In [39]:
```
model.fit(x_train, y_train, epochs = 100, batch_size = 32)
```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 12)	0
dense (Dense)	(None, 256)	3328
dense_1 (Dense)	(None, 128)	32896
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 32)	2080
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 8)	136
dense_6 (Dense)	(None, 1)	9

Total params: 47,233
Trainable params: 47,233
Non-trainable params: 0

```
In [39]: model.fit(x_train, y_train, epochs = 100, batch_size = 32)
```

- As we notice , among the three models we find the ANN model to be the most accurate on the dataset.
- Next we visualize the output of the three models graphically, where the original output is represented in blue colour, the svm output is represented in green colour and the ANN output is represented in red colour.



IMPLEMENTATION AND OUTPUT

- To implement and check the accurate output of the model, we first declare variables for each attribute of the dataset and provide user defined values to the variables.
- These values are fed as input to the model which predicts the final output, i.e, the price of the house based on the given input.
- The final output or prediction is as follows-

In [56]:
```
housing_median_age = 54
total_rooms = 500
total_bedrooms = 200
population = 200
households = 200
median_income = 3.5
affordable = 1
ocean_proximity_1 = 0
ocean_proximity_INLAND = 0
ocean_proximity_ISLAND = 1
ocean_proximity_NEARBAY = 0
ocean_proximity_NEAROCEAN = 0

input = [housing_median_age,total_rooms,total_bedrooms,population,households,median_income,affordable,ocean_proximity_1,ocean_pr

input = np.array(input).reshape(1,-1)

output = model.predict(input)
print("the house price is:",output[0][0])
```

```
the house price is: 216358.81
```

CONCLUSION

From the training and testing of dataset on the model, a strong deduction can be made that the model ANN works most effectively on the data producing lower levels of errors, and hence we can conclude that Deep Learning techniques prove useful in the implementation and estimation of HOUSE PRICE PREDICTION.

THANK YOU