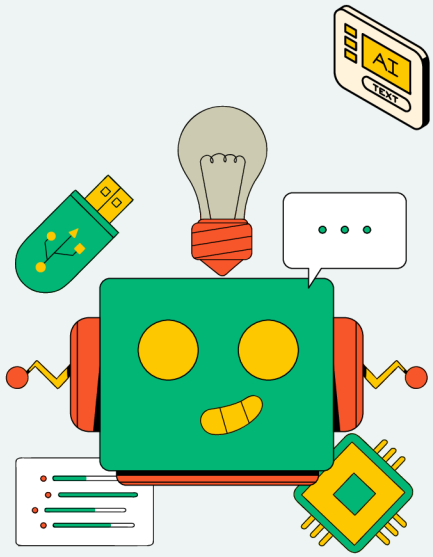
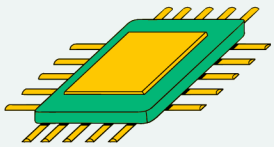


MACHINE LEARNING PROJECT

PRESENTATION

PRESENTED BY:

JANVI AGRAWAL
MEHUL RAI



INTRODUCTION

REAL ESTATE – PRICE PREDICTION

This project focuses on predicting real estate prices using machine learning techniques. The goal is to develop a model that estimates property values based on various factors such as the number of rooms, location, crime rate, and more. Using a dataset like the Boston Housing Dataset, the project aims to train a regression model to predict house prices with high accuracy.

OBJECTIVE OF THE PROJECT:

- Analyze factors affecting property prices.
- Develop a predictive model using machine learning.
- Evaluate model performance for accuracy.
- Provide insights to help in real estate decision-making.
- Create a deployable tool for real-time price predictions.

MODEL TO BUILD

01

SUPERVISED LEARNING

OUR MODEL IS SUPERVISED LEARNING MODEL BECAUSE WE HAVE LABELS AND FEATURES ARE PRESENT

02

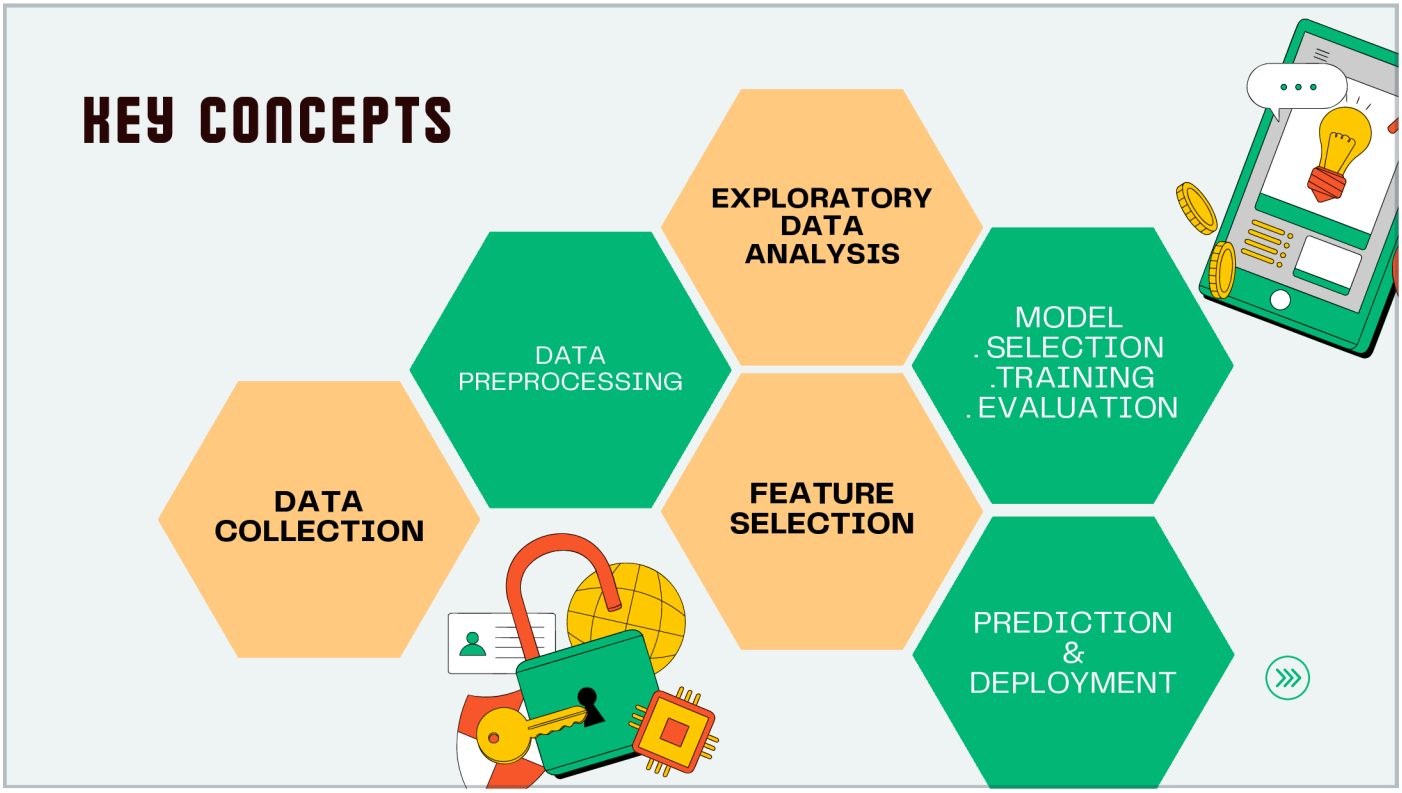
REGRESSION TASK

OUR MODEL IS REGRESSION TASK BECAUSE OUR GOAL IS TO PREDICT A CONTINOUS VALUE

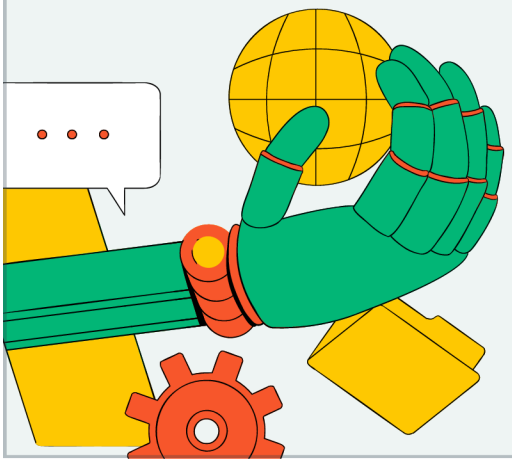
03

BATCH LEARNING

OUR MODEL IS USING BATCH LEARNING BECAUSE WE HAVE A SET OF DATA WHICH WE WILL USE TO TRAIN AND TEST OUR MODEL


KEY CONCEPTS
EXPLORATORY DATA ANALYSIS
DATA PREPROCESSING
MODEL
. SELECTION
. TRAINING
. EVALUATION
DATA COLLECTION
FEATURE SELECTION
PREDICTION & DEPLOYMENT

DATASET FOR THE MODEL



A sample Dataset of a housing company is used in the model.

our dataset contains 14 attributes(features)
13 continuous attributes and 1 – binary value attribute.

RANGE INDEX: 506 ENTRIES, 0 TO 505.

```
housing.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

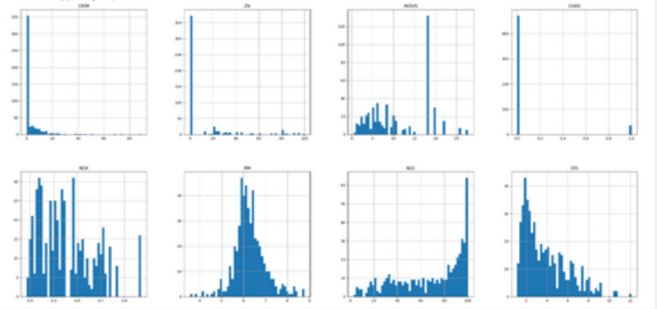
In [7]:
```
housing.describe()
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000

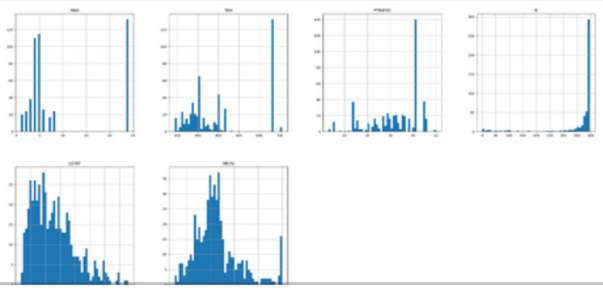
```
In [8]: housing.hist(bins=50, figsize=(30,30))

Out[8]: array([[<Axes: title={'center': 'CRIM'}>, <Axes: title={'center': 'ZN'}>,
                <Axes: title={'center': 'INDUS'}>,
                <Axes: title={'center': 'CHAS'}>],
               [<Axes: title={'center': 'NOX'}>, <Axes: title={'center': 'RM'}>,
                <Axes: title={'center': 'AGE'}>, <Axes: title={'center': 'DIS'}>],
               [<Axes: title={'center': 'RAD'}>, <Axes: title={'center': 'TAX'}>,
                <Axes: title={'center': 'PTRATIO'}>,
                <Axes: title={'center': 'B'}>],
               [<Axes: title={'center': 'LSTAT'}>,
                <Axes: title={'center': 'MEDV'}>, <Axes: >, <Axes: >]],
              dtype=object)
```



Bins: A value describing the number of divisions the graph is to be divided in.

FigSize: The value describing the length of x and y axis.



```
Correlation Matrix:
             CRIM        ZN     INDUS      CHAS       NOX        RM       AGE
CRIM     1.000000 -0.200469  0.406583 -0.055892  0.420972 -0.219247  0.352734
ZN      -0.200469  1.000000 -0.533828 -0.042697 -0.516604  0.311991 -0.569537
INDUS    0.406583 -0.533828  1.000000  0.062938  0.763651 -0.391676  0.644779
CHAS    -0.055892 -0.042697  0.062938  1.000000  0.091203  0.091251  0.086518
NOX      0.420972 -0.516604  0.763651  0.091203  1.000000 -0.302188  0.731470
RM      -0.219247  0.311991 -0.391676  0.091251 -0.302188  1.000000 -0.240265
AGE      0.352734 -0.569537  0.644779  0.086518  0.731470 -0.240265  1.000000
DIS     -0.379670  0.664408 -0.708027 -0.099176 -0.769230  0.205246 -0.747881
RAD      0.625505 -0.311948  0.595129 -0.007368  0.611441 -0.209847  0.456022
TAX      0.582764 -0.314563  0.720760 -0.035587  0.668023 -0.292048  0.506456
PTRATIO  0.289946 -0.391679  0.383248 -0.121515  0.188933 -0.355501  0.261515
B       -0.385064  0.175520 -0.356977  0.048788 -0.380051  0.128069 -0.273534
LSTAT    0.455621 -0.412995  0.603800 -0.053929  0.590879 -0.613808  0.602339
MEDV    -0.388305  0.360445 -0.483725  0.175260 -0.427321  0.695360 -0.376955

              DIS       RAD       TAX   PTRATIO         B     LSTAT      MEDV
CRIM    -0.379670  0.625505  0.582764  0.289946 -0.385064  0.455621 -0.388305
ZN       0.664408 -0.311948 -0.314563 -0.391679  0.175520 -0.412995  0.360445
INDUS   -0.708027  0.595129  0.720760  0.383248 -0.356977  0.603800 -0.483725
CHAS    -0.099176 -0.007368 -0.035587 -0.121515  0.048788 -0.053929  0.175260
NOX     -0.769230  0.611441  0.668023  0.188933 -0.380051  0.590879 -0.427321
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808  0.695360
AGE     -0.747881  0.456022  0.506456  0.261515 -0.273534  0.602339 -0.376955
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996  0.249929
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.488676 -0.381626
TAX     -0.534432  0.910228  1.000000  0.460853 -0.441808  0.543993 -0.468536
PTRATIO -0.232471  0.464741  0.460853  1.000000 -0.177383  0.374044 -0.507787
B        0.291512 -0.444413 -0.441808 -0.177383  1.000000 -0.366087  0.333461
LSTAT   -0.496996  0.488676  0.543993  0.374044 -0.366087  1.000000 -0.737663
MEDV     0.249929 -0.381626 -0.468536 -0.507787  0.333461 -0.737663  1.000000
```

```python
# Correlation matrix
correlation_matrix = housing.corr(numeric_only=True)
print("\nCorrelation Matrix:\n", correlation_matrix)
```

Correlation Matrix: A correlation matrix is a table showing the correlation coefficients between variables in a dataset. It measures how strongly two variables are related to each other. Values range from:

- +1: Perfect positive correlation (as one variable increases, the other also increases).
- 0: No correlation (the variables are independent).
- −1: Perfect negative correlation (as one variable increases, the other decreases).

Train-Test Splitting

In [12]:

```
import numpy as np

def split_train_test(data, test_ratio):
    shuffled = np.random.permutation(len(data))
    print(shuffled)
    test_set_size = int(len(data)*test_ratio)
    test_indices = shuffled[:test_set_size]
    train_indices = shuffled[test_set_size:]
    return data.iloc[train_indices],data.iloc[test_indices]
```

In [13]:

```
train_set, test_set = split_train_test(housing, 0.2)
```

```
[452 390   2 160 186  97  66 307 337 220 346  23 377 332 245 426 425 262
 340   1 104 105 395 419 257 331 132 166 241 298 115 252 443  29  40 433
 315 454 458 265 113  99 305  92 360 497 209  62 414 174  27 283 163 126
 199  30  54 429 229   3 190   0 203 463 167 444 243 244 215 261 196 445
  88 398 446 147 434 114 498 343 130 482 448 211 117 372 435 375 251  12
  75  76  74 237 231 327   8 308  64 234 365 266 228 465 353 405  81 396
 453  42 326 362 136 431 491 131   9 361 503 222 352 504  71  10  37 263
 399  78 329 407 191 388  48 447 256 304  50  31  25 492 212 159 302 289
 392 351 194 485 496 488 204 139  53 386  96 249 489 238 319 232 427 450
 281 155 476 189 172 461 290 200  72 413 227  16  19  45 311 106 347 478
 235 344 260  18 437  80 164 233 424 156 350  35 107 247 279 472 250 168
  90  13 140 384 125 438  87 142 210   4 187  44 409 323  28 479 466 275
 138 338 277 282 487  14 376 122 462 103 342 300 358  41 258 379 505 195
  98 371   7 221 356 182 207 422 146 318 201 459 313 180 303 224  47 310
 322 469 185 225 370 328 354 321 494 202  43 216 397 501   6  91 100 292
  38 165  22 309 101 341 471 394 449 213 284 408 473 171 367 393 441  73
 339 411 480 226 230 223 481 161 502 173 288 248  70  11 170  46 272 455
 389 184 320 120 325 278 144 316 439 133 374 192 333  77 483 137 420 162
 148 178 177 317  57 295 400   5 154 112  83  60  94 286  15 312 102 145
 287 330 157 373  95 141  63 274 474 134 175 118 151 214 348 470 345  52
 169 181 271 267 335 349 475 205 432 280 259 440 416 334 477 293 336 285
```

```
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}")
```

```
Rows in train set: 405
Rows in test set: 101
```

In [15]:

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}\n")
```

```
Rows in train set: 404
Rows in test set: 102
```

Training set: The data used to train the model
Test set: the data used to test how well the model performs after training.

The dataset is split into 80-20 ratio
random_state(42): Is used to maintain a fix split according to the domain of the data

Thank you