Tribhuvan University

Prithivi Narayan Campus

त्रिभुवन विश्वविद्यालय
पृथ्वीनारायण क्याम्पस

Lab Report Of Artificial Intelligence

Subject Code : CSC2625

Submitted To:

Department of Bsc.CSIT

Lecturer: MR. Thakur Upadya

Prithivi Narayan Campus

Submitted By:

Sulav Baral

Roll No: 09

Symbol No: 80012181

INDEX

S. N	Title	Submit Date	Sign
1.	Inference with logical deductions using swi-prolog	2082/05/	
2.	Implementing Classification Algorithm using Naïve Bayes	2082/05/	
3.	Implementing a simple backpropagation algorithm using python.	2082/05/	
4.	Implementing a regression model to predict housing prices.	2082/05/	

Lab: 1

TITLE: INFERENCE WITH LOGICAL DEDUCTIONS USING SWI-PROLOG

OBJECTIVES

- To understand how inference and logical reasoning are carried out using Prolog.
- To explore how facts, rules, and queries form the basis of a knowledge base.
- To perform logical queries and test inference mechanisms on the created knowledge base using SWI-Prolog.

THEORY

Artificial Intelligence (AI) is largely built on the idea of reasoning and decision-making through logical inference. One of the earliest and most fundamental approaches to AI is symbolic reasoning, where knowledge about the world is represented using symbols and relationships, and conclusions are drawn from these representations through logical deduction. Logic programming, particularly Prolog (Programming in Logic), provides a direct way to express such knowledge bases and query them systematically.

In Prolog, a knowledge base (KB) is composed of facts and rules. Facts represent basic information that is unconditionally true in the system. For example, girl(priya). asserts that Priya is a girl. Rules, on the other hand, describe conditional relationships between facts. For instance, likes(priya, jaya) :- can_cook(jaya). states that Priya likes Jaya if Jaya can cook. These rules reflect the use of implication in logic, where the truth of one fact depends on the truth of another.

The process of inference in Prolog is handled by its internal reasoning engine through a mechanism called backward chaining. When a query is asked, Prolog attempts to satisfy it by searching through facts and applying rules, tracing backwards to check if conditions hold true. This search strategy enables Prolog to answer complex queries by combining simple facts. For example, asking ?- can_cook(X). allows Prolog to infer all entities X for which the fact can_cook(X) is true.

Logical inference is powerful because it mirrors human-like reasoning. For instance, from simple facts such as “Lili dances” and a rule “Lili is happy if she dances,” Prolog can conclude that Lili is happy. Similarly, combining facts and conditions allows us to describe more complex real-world situations like friendship, hunger, or the decision to go out and play. This structured approach to reasoning is the cornerstone of many AI applications, from expert systems to knowledge representation and reasoning systems.

Overall, Prolog demonstrates how symbolic logic can be operationalized to build small-scale intelligent systems. While it may not be as fast as modern machine learning methods, it provides clarity in understanding how machines can reason using structured logical rules.

Syntax of proglog:

Using the following truth-functional symbols, the Prolog expressions are

English	Predicate Calculus	Prolog
If	-->	:-
Not	~	Not
Or	V	;
and	^	,

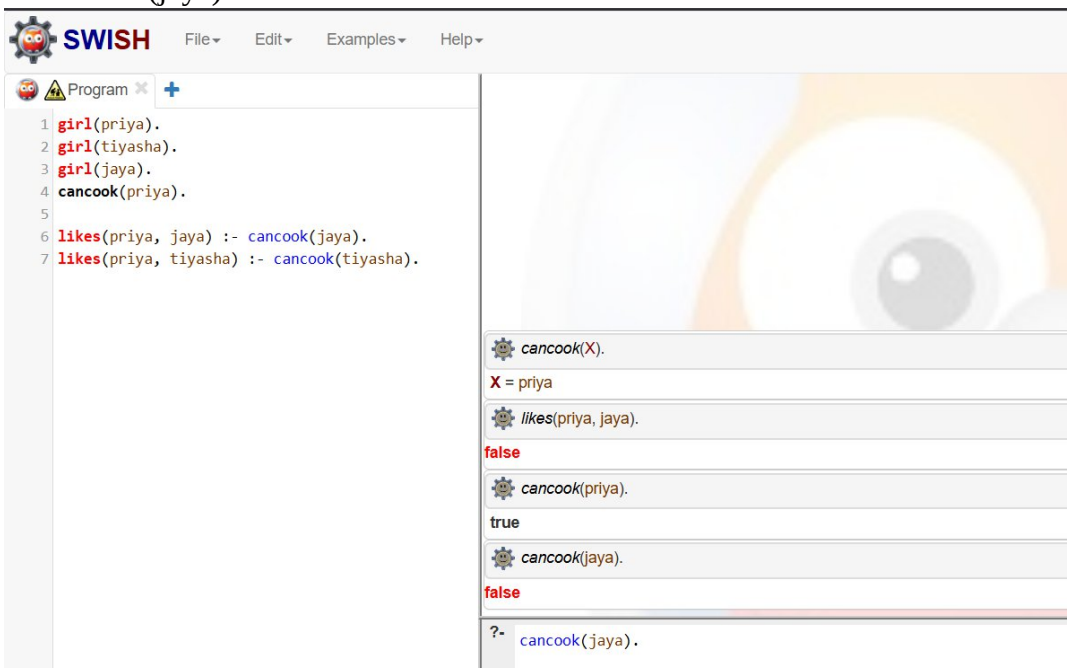
comprised. These symbols have the same interpretation as in the predicate calculus.

DEMONSTRATION

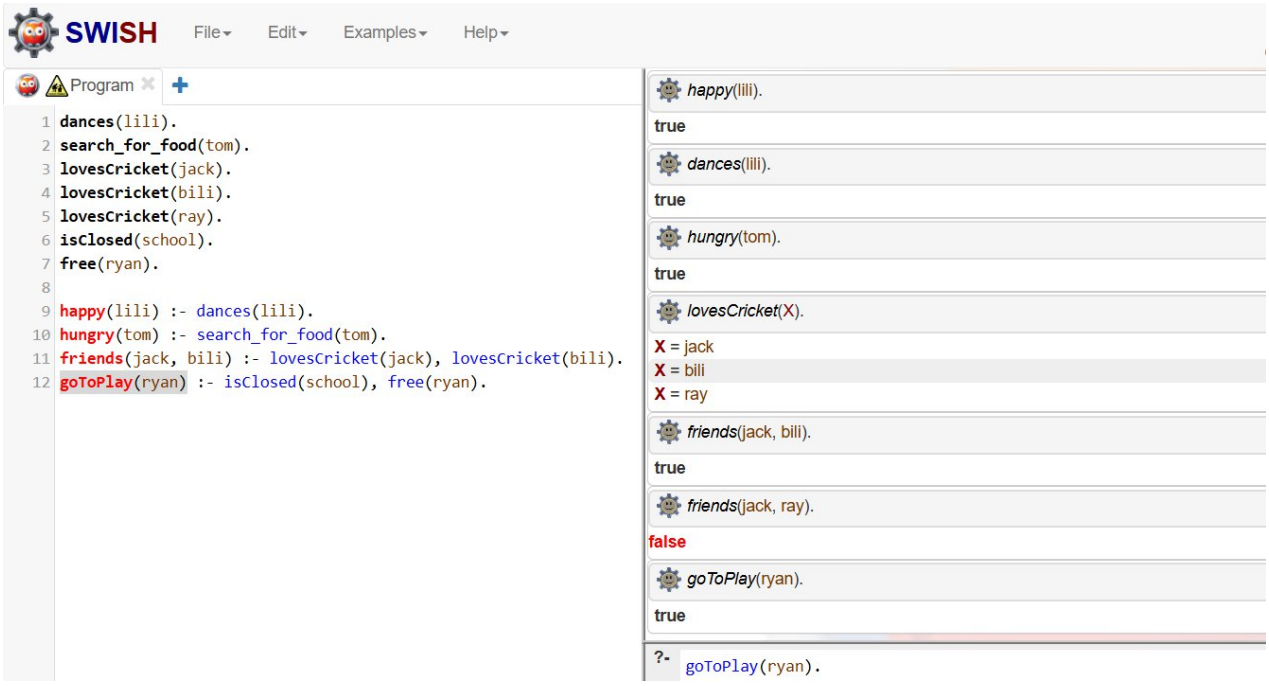
Facts :
girl(priya).
girl(tiyasha).
girl(jaya).
can_cook(priya).

Rules:
likes(priya, jaya) :- can_cook(jaya).
likes(priya, tiyasha) :- can_cook(tiyasha).

Example Queries
- can_cook(X).
- likes(Priya, jaya).
- cancook(Priya).
- cancook(jaya).



Additional Knowledge Base



RESULT AND DISCUSSION

In this experiment, we successfully created a simple knowledge base in Prolog consisting of **facts and rules**. By running queries, we observed how Prolog applies **logical inference** to provide answers. For instance, the query ?- can_cook(X). correctly identified Priya as the one who can cook. Similarly, when conditions were satisfied, Prolog was able to infer conclusions like “Lili is happy if she dances.” This demonstrates the fundamental reasoning ability of Prolog, where the system does not store precomputed answers but **derives solutions dynamically based on logical relationships**. The experiment also highlighted how more complex rules can be built to represent real-world knowledge, such as defining friendship based on shared interests or deciding to play based on conditions like school being closed and having free time. Through these examples, it becomes clear how **symbolic logic-based AI** works by encoding knowledge in structured form and then applying inference rules to draw meaningful conclusions. This lays the foundation for understanding larger AI systems that rely on reasoning, such as **expert systems, planning systems, and intelligent agents**.

CONCLUSION

The experiment demonstrated how **Prolog and logical inference** can be used to represent knowledge and perform deductions. By building a small knowledge base and testing queries, we gained an understanding of how facts and rules interact in Prolog. This provides valuable insight into **symbolic AI**, where knowledge is explicitly represented and manipulated, as opposed to modern statistical learning approaches. Overall, the experiment reinforced the importance of logic programming in the history and foundations of Artificial Intelligence.

Lab: 2

TITLE: IMPLEMENTING CLASSIFICATION ALGORITHM USING NAÏVE BAYES

OBJECTIVES

- To understand the working principle of the Naïve Bayes classification algorithm.
- To apply Naïve Bayes on a dataset and analyze results using Excel.
- To observe how conditional probabilities are used for classification tasks.

THEORY

Classification is a fundamental problem in Artificial Intelligence (AI) and Machine Learning, where the goal is to categorize data into predefined classes based on given features. Among various classification algorithms, **Naïve Bayes** is one of the simplest yet highly effective methods, particularly for text classification, spam detection, and medical diagnosis.

The Naïve Bayes classifier is based on **Bayes' Theorem**, which describes the probability of a hypothesis HHH given some observed evidence EEE:

$$[P(E) = \frac{P(H) \cdot P(H)}{P(E)}]$$

Here,

- $P(E)$ is the posterior probability of hypothesis HHH after seeing evidence EEE.
- $P(H)$is the likelihood of evidence given that the hypothesis holds.
- $P(H)$ is the prior probability of the hypothesis.
- P(E) is the probability of the evidence.

In classification, the hypothesis corresponds to a class label, and the evidence corresponds to the feature values of the instance to be classified. The classifier assigns a new instance to the class with the **maximum posterior probability**.

The assumption made by Naïve Bayes is that all features are **conditionally independent** given the class label. Although this assumption is rarely true in practice, the algorithm often performs remarkably well, even on datasets with dependent features.

In this lab, Naïve Bayes was implemented using an Excel sheet. The dataset contained features and class labels, and we computed prior probabilities, likelihood probabilities, and posterior probabilities to classify instances. The Excel formulas helped automate these calculations, making it easier to observe how each probability contributes to the final decision.

Naïve Bayes is especially useful because of its simplicity, efficiency, and ability to handle large feature spaces. However, it struggles when features are highly correlated or when continuous variables do not follow the assumed distribution (commonly Gaussian).

DEMONSTRATION

Final Result: The instance is classified as **Yes** because posterior probability of "Yes" is higher.
(Given fig below:)

SN	Outlook	Temperature	Humidity	Wind	Play
1	Rainy	Hot	High	No	No
2	Rainy	Hot	High	Yes	No
3	Overcast	Hot	High	No	Yes
4	Sunny	Mild	High	No	Yes
5	Sunny	Cool	Normal	No	Yes
6	Sunny	Cool	Normal	Yes	No
7	Overcast	Cool	Normal	Yes	Yes
8	Rainy	Mild	High	No	No
9	Rainy	Cool	Normal	No	Yes
10	Sunny	Mild	Normal	No	Yes
11	Rainy	Mild	Normal	Yes	Yes
12	Overcast	Mild	High	Yes	Yes
13	Overcast	Hot	Normal	No	Yes
14	Sunny	Mild	High	No	No
15	Rainy	Hot	High	Yes	Yes
16					
17					
18					
19					
20					
21					
22					
23					

Outlook	Play=Yes	Play=No
Rainy	3	3
Sunny	3	2
Overcast	4	0
Coltotal	10	5

	P(Yes/Outlooki)	P(No/Outlooki)
	0.3000	0.6000
	0.4000	0.4000
	0.4000	0.0000
	1.0000	1.0000

Temparature	Play=Yes	Play=No
Hot	3	2
Mild	4	2
Cool	3	1
Coltotal	10	5

	P(Yes/Tempi)	P(No/Tempi)
	0.3000	0.4000
	0.4000	0.4000
	0.3000	0.2000
	1.0000	1.0000

Humidity	Play=Yes	Play=No
High	4	4
Normal	6	1
Coltotal	10	5

	P(Yes/Humidi)	P(No/Humid)
	0.4000	0.8000
	0.6000	0.2000
	1	1

Wind	Play=Yes	Play=No
Yes	4	2
No	6	3
Coltotal	10	5

	P(Play=Yes)	P(Play=No)
	0.6666667	0.333333333

Prediction for play or not

Outlook	Temperature	Humidity	Wind	Posterior/ Play=Yes	Posterior/ Play=No	Normalized P	Prediction of play
Rainy	Hot	High	Yes	0.0096	0.0256	0.27273	No
Rainy	Hot	High	No				
Rainy	Hot	Normal	Yes				
Rainy	Hot	Normal	No				
Sunny	Hot	High	No				
Rainy	Hot	High	Yes				
Sunny	Hot	Normal	No	0.0216	0.0064	0.77143	Yes

Steps followed in Excel:

1. **Data Preparation** – The dataset was provided with input features and class labels.
2. **Prior Probabilities** – Computed by dividing the frequency of each class by the total number of records.
3. **Likelihood Probabilities** – For each feature value, conditional probabilities given a class were calculated.
4. **Posterior Probabilities** – Using Bayes' theorem, posterior probabilities were computed for each class.
5. **Classification Decision** – The class with the highest posterior probability was assigned to the new data point.

RESULT AND DISCUSSION

The Naïve Bayes algorithm was successfully applied to the dataset using Excel. By calculating prior, likelihood, and posterior probabilities step by step, we observed how classification decisions are made mathematically. The results confirmed that the algorithm assigns each instance to the class with the highest posterior probability.

One of the key observations was how simple multiplication of probabilities can effectively capture the relationship between features and classes.

CONCLUSION

This experiment demonstrated the implementation of the Naïve Bayes classification algorithm using Excel. Through probability-based calculations, we understood how Naïve Bayes makes classification decisions. The lab highlighted the importance of Bayes' theorem in AI and showed how a simple probabilistic model can achieve effective classification.

Lab: 3
TITLE: IMPLEMENTING SIMPLE BACKPROPAGATION ALGORITHM USING PYTHON

OBJECTIVES

- To understand the working of the backpropagation algorithm in neural networks.
- To implement a simple neural network using Python and NumPy.
- To observe how weights are updated during training and how the network output approaches the target value.

THEORY

Artificial Neural Networks (ANNs) form the basis of many modern AI applications, including image recognition, natural language processing, and autonomous systems. They mimic the functioning of the human brain by organizing artificial neurons into layers and passing information forward through weighted connections. The strength of these connections (weights) determines the response of the network to given inputs.

Training a neural network requires an algorithm that can adjust the weights so that the network's output matches the expected target. The **Backpropagation Algorithm** is the standard technique for this task. It is essentially a supervised learning method that uses **gradient descent** to minimize the error between the predicted output and the actual target. The process involves two main stages:

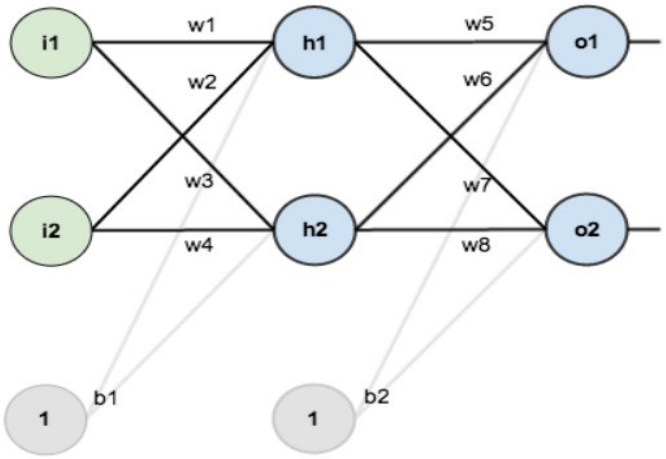
1. **Forward Propagation:** Input values are passed through the network, layer by layer. Each neuron applies a weighted sum followed by an activation function (in this case, the sigmoid function). This produces the predicted output.
2. **Backward Propagation:** The error is calculated as the difference between the predicted output and the target output. Using the **chain rule of differentiation**, this error is propagated backward through the network. Gradients of the error with respect to each weight are computed, and weights are updated in the opposite direction of the gradient to reduce the error.

In this experiment, the network consisted of an **input layer (2 neurons), one hidden layer (2 neurons), and an output layer (1 neuron)**. A sigmoid activation function was used, which squashes values into the range (0,1), making it suitable for small-scale examples. Unlike typical implementations with random initialization, here the weights were initialized with fixed values so that updates could be easily traced. This made it possible to observe clearly how weights were modified across training epochs.

Backpropagation is powerful because it can generalize to any differentiable activation function and any network depth. It is the foundation of deep learning, enabling neural networks to learn complex, non-linear decision boundaries. However, it also comes with challenges like sensitivity to learning rates, risk of getting stuck in local minima, and the vanishing gradient problem in deeper networks. Despite these issues, backpropagation remains the most fundamental learning algorithm for feedforward neural networks.

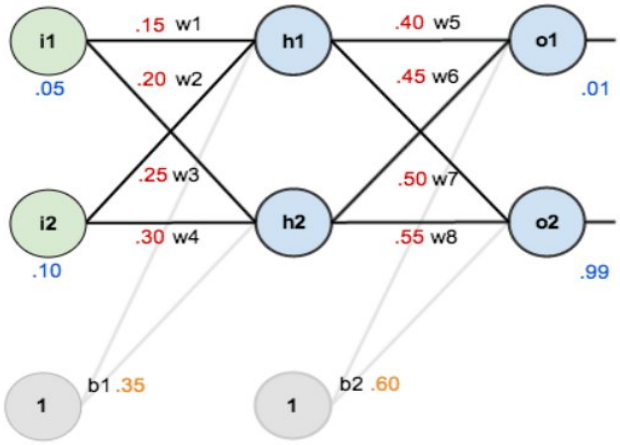
DEMONSTRATION

Basic structure:



\

initial weights, the biases, and training inputs/outputs:



predicted output in epoch 1 :

```
predicted output in epoch 0 = [[0.69084195]]
inout to hidden weights : [[0.09897073 0.79727456]
 [0.39735329 0.59299172]]
hidden to output weights : [[0.27564654]
 [0.871703  ]]

Input: [[0.35 0.9 ]]
Predicted Output after training: [[0.6831607]]
Target Output: [[0.5]]
```

predicted output in epoch 50 :

```
predicted output in epoch 48 = [[0.51782512]]
inout to hidden weights : [[0.09664301 0.75950641]
 [0.39136775 0.49587362]]
hidden to output weights : [[-0.22390468]
 [ 0.29963823]]
predicted output in epoch 49 = [[0.5169235]]
inout to hidden weights : [[0.0967228  0.75940855]
 [0.39157291 0.49562198]]
hidden to output weights : [[-0.22642057]
 [ 0.29680312]]

Input: [[0.35 0.9 ]]
Predicted Output after training: [[0.51606732]]
Target Output: [[0.5]]
```

predicted output in epoch 100:

```
predicted output in epoch 98 = [[0.50132398]]
inout to hidden weights : [[0.09824704 0.75786559]
 [0.39549238 0.49165438]]
hidden to output weights : [[-0.26995484]
 [ 0.24782473]]
predicted output in epoch 99 = [[0.50125686]]
inout to hidden weights : [[0.09825418 0.75785957]
 [0.39551076 0.49163888]]
hidden to output weights : [[-0.27014223]
 [ 0.24761423]]

Input: [[0.35 0.9 ]]
Predicted Output after training: [[0.50119314]]
Target Output: [[0.5]]
```

RESULT AND DISCUSSION

The experiment successfully demonstrated the implementation of the **backpropagation algorithm** in Python using a class-based neural network model. Starting from initial fixed weights, the network performed forward propagation to compute predictions and then used backward propagation to adjust weights based on the observed error. During training, the predicted output gradually shifted closer to the target output (0.5). By the end of 100 epochs, the predicted output was approximately **0.502**, which is very close to the expected target.

CONCLUSION

In this experiment, we implemented a **simple neural network with backpropagation** using Python and NumPy. Through forward and backward passes, the network adjusted its weights and successfully minimized the error between predicted and target outputs. This provided a clear understanding of the mechanics of backpropagation, reinforcing its importance as the foundation of training neural networks in AI.

Lab: 4
TITLE: IMPLEMENTING REGRESSION MODEL TO PREDICT HOUSING PRICES.

OBJECTIVES

- To understand how regression can be applied to real-world problems for predictive modeling.
- To build and evaluate a regression model for predicting housing prices using the California Housing dataset.
- To analyze correlations between housing features and median house value.
- To implement regression using Python libraries such as Pandas, scikit-learn, and Statsmodels.

THEORY

Artificial Intelligence (AI) not only deals with symbolic reasoning and problem-solving but also with learning from data and making predictions. One of the fundamental techniques in this domain is **regression analysis**, which belongs to the class of **supervised machine learning algorithms**. Regression is widely used when the target variable is continuous and the objective is to model the relationship between independent variables (features) and the dependent variable (output).
In this experiment, we focus on **linear regression**, which assumes that the dependent variable can be expressed as a linear combination of the independent variables. Mathematically, it can be expressed as:

$$[y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon]$$

Here,

- y = predicted output (median house value),
- β_0 = intercept,
- $\beta_1, \beta_2, \beta_3 \ldots\ldots \beta_n$ = coefficients representing the effect of each feature,
- ϵ = error term.

The **California housing dataset** contains attributes such as longitude, latitude, median income, housing age, total rooms, total bedrooms, population, and households. The target variable is **median_house_value**, which we want to predict. Before applying regression, it is essential to handle **missing values**, split the dataset into training and test subsets, and explore correlations to understand which features strongly influence the target.
To evaluate the model, we use statistical measures such as the **R-squared value (R2R^2R2)**, which indicates how much of the variance in the dependent variable is explained by the independent variables. A higher R2R^2R2 means the model fits the data better. Additionally, analyzing **residuals** (differences between actual and predicted values) is critical to check whether the assumptions of regression (linearity, homoscedasticity, independence, and normality of errors) are reasonably satisfied.
This lab also uses **Statsmodels**, which provides detailed regression summaries, including p-values, confidence intervals, and adjusted R2R^2R2, giving deeper insights into the significance of each feature. By completing this experiment, we demonstrate how regression can be applied in practice to predict housing prices and how statistical analysis supports decision-making in AI and Data Science.

DEMONSTRATION

California Housing Dataset:

```
[2] housing = pd.read_csv('/content/sample_data/california_housing_train.csv')
    housing.shape
```

```
(17000, 9)
```

```
housing.head()
```

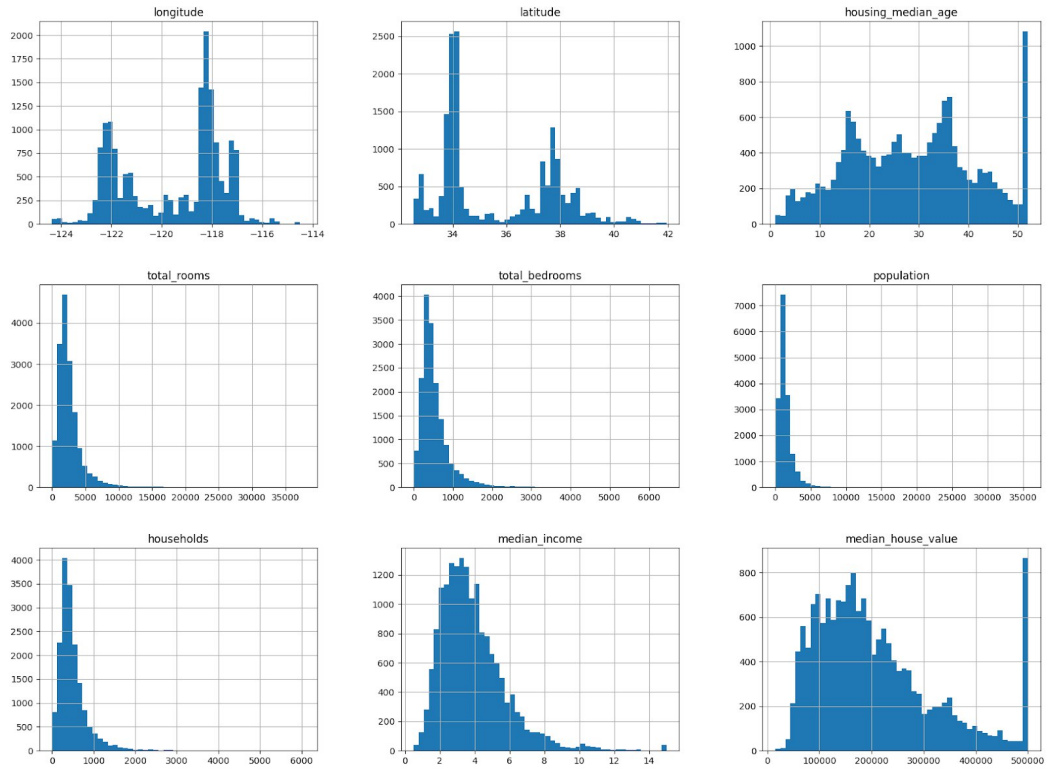
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0	1.4936	66900.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0	1.8200	80100.0
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.0	1.6509	85700.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0	3.1917	73400.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.0	1.9250	65500.0

Next steps: Generate code with housing | View recommended plots | New interactive sheet

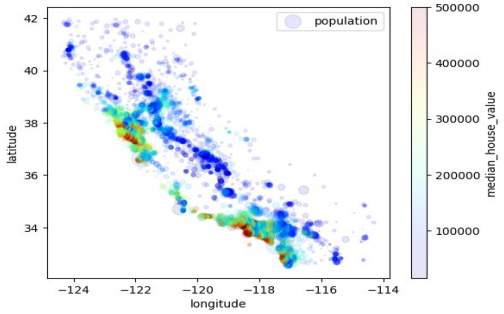
Housing describes:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000
mean	-119.562108	35.625225	28.589353	2643.664412	539.410824	1429.573941	501.221941	3.883578	207300.912353
std	2.005166	2.137340	12.586937	2179.947071	421.499452	1147.852959	384.520841	1.908157	115983.764387
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.790000	33.930000	18.000000	1462.000000	297.000000	790.000000	282.000000	2.566375	119400.000000
50%	-118.490000	34.250000	29.000000	2127.000000	434.000000	1167.000000	409.000000	3.544600	180400.000000
75%	-118.000000	37.720000	37.000000	3151.250000	648.250000	1721.000000	605.250000	4.767000	265000.000000
max	-114.310000	41.950000	52.000000	37937.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

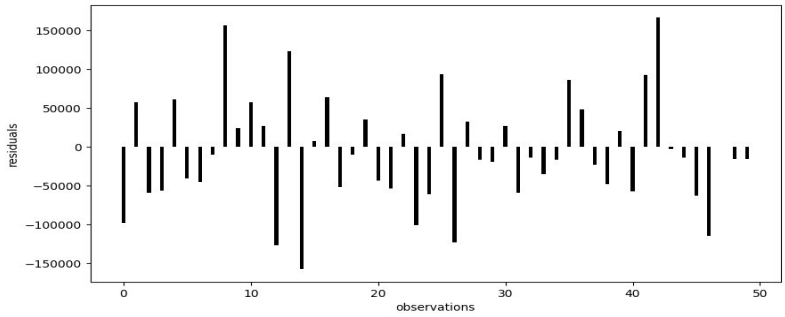
Histogram Information:



Regression graph:



Residuals vs Observations:



Summary:

```
nicer_OLS=sm.OLS(y_train,X_train).fit()
nicer_OLS.summary()
```

OLS Regression Results

Dep. Variable:	median_house_value	R-squared:	0.640
Model:	OLS	Adj. R-squared:	0.640
Method:	Least Squares	F-statistic:	3018.
Date:	Sat, 06 Sep 2025	Prob (F-statistic):	0.00
Time:	21:15:38	Log-Likelihood:	-1.7091e+05
No. Observations:	13600	AIC:	3.418e+05
Df Residuals:	13591	BIC:	3.419e+05
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.674e+06	7.73e+04	-47.509	0.000	-3.83e+06	-3.52e+06
longitude	-4.374e+04	882.049	-49.586	0.000	-4.55e+04	-4.2e+04
latitude	-4.342e+04	832.932	-52.125	0.000	-4.5e+04	-4.18e+04
housing_median_age	1099.6142	53.209	20.666	0.000	995.317	1203.911
total_rooms	-9.1021	0.962	-9.457	0.000	-10.989	-7.216
total_bedrooms	116.1846	8.598	13.513	0.000	99.331	133.038
population	-36.7430	1.299	-28.277	0.000	-39.290	-34.196
households	46.1878	9.477	4.874	0.000	27.611	64.764
median_income	4.076e+04	413.949	98.475	0.000	4e+04	4.16e+04

Omnibus:	3169.852	Durbin-Watson:	1.985
Prob(Omnibus):	0.000	Jarque-Bera (JB):	12079.465
Skew:	1.124	Prob(JB):	0.00
Kurtosis:	7.033	Cond. No.	5.15e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.15e+05. This might indicate that there are strong multicollinearity or other numerical problems.

RESULT AND DISCUSSION

The regression model was successfully implemented and tested on the California housing dataset. The results revealed that **median income** was the most influential feature in predicting house prices, while geographical attributes like longitude and latitude also contributed to price variation. The model achieved an R2R^2R2 score of approximately 0.64, indicating that about 64% of the variance in house prices was explained by the chosen features.

Residual plots showed that most prediction errors were randomly distributed, suggesting that the linear regression assumptions were mostly valid. However, slight deviations hinted at the presence of non-linear relationships that the linear model could not fully capture.

CONCLUSION

This experiment successfully demonstrated how regression can be applied to predict housing prices using real-world datasets. By building and training a linear regression model, analyzing residuals, and interpreting statistical summaries, we gained insights into both the predictive power and limitations of linear regression. The lab reinforced the role of regression in AI and Data Science, bridging statistical analysis with machine learning for practical problem-solving.