GRID157 - M02W03 - Statistics

1. Introduction

In the era of artificial intelligence, data isn't just the fuel powering our models — it's also the compass that guides their decisions. But before machine learning algorithms can truly *learn* from data, we — as model designers — must first understand the structure, patterns, and relationships within it. That's where statistics plays a critical role.

Imagine working with real-world datasets like housing prices, where features include variables such as area, number of rooms, age of the house, distance to the city center, and crime rate. These datasets are often messy: missing values, outliers, duplicate information, or misleading correlations can quietly degrade model performance if left untreated.

In this blog, we'll walk through *essential statistical concepts* **and data preprocessing techniques** that lay the foundation for accurate modeling. You'll learn not only what *mean*, *median*, *variance*, and *standard deviation* are — but also how to use them to spot inconsistencies or unusual patterns. We'll explore how to detect and handle *missing values* and **outliers** using pandas and numpy, and most importantly, how to analyse *relationships between variables* through **covariance** and **correlation coefficients**.

Using synthetic and real-world-like examples, we'll show how to compute and visualise a **correlation matrix** using Python. With tools like **heat maps**, you'll gain practical insight into which features are most relevant for your models — and which ones might be dropped or transformed.

By the end of this blog, you'll be equipped to:

- Clean and prepare tabular data using pandas and numpy
- Apply statistical metrics to summarize and understand distributions
- Handle missing data and detect outliers systematically
- Compute and interpret covariance and Pearson correlation
- Use heat maps and correlation matrices to guide feature selection

Let's begin by building a strong foundation in statistics — one that connects raw numbers to real-world insights and smarter machine learning models.

2. Statistics' concepts

2.1 Mean, Median, Variance, and Standard Deviation

Here are the definitions, mathematical formulas, and specific examples for Mean, Median, Variance, and Standard Deviation, based on the scores of 10 students.
For example:
A class has 10 students with the following test scores (already sorted in ascending order):

`scores = [3, 4, 5, 6, 6, 7, 8, 8, 9, 10]`

Mean

- **Definition:** The Mean is the sum of all values divided by the number of elements. It is a common indicator to describe the central tendency of data.
- Formula:

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Where:

- x_i is the score of the i-th student
- n is the total number of students

Example Application:

$$\text{Mean} = \frac{3+4+5+6+6+7+8+8+9+10}{10} = \frac{66}{10} = 6.6$$

✅
Meaning: The average student scored 6.6 points. This is the "balancing" point of the class's scores.

Median

Definition: The Median is the middle value of a dataset after it has been sorted in ascending order. If the number of elements is even, the median is the average of the two middle numbers.

Formula:

- If n is odd:

$$\text{Median} = x_{\frac{n+1}{2}}$$

- If n is even:

$$\text{Median} = \frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2}$$

Example Application:
The dataset has 10 elements (even), so the two middle numbers are at the 5th and 6th positions:

$$\text{Median} = \frac{6+7}{2} = 6.5$$

✅

Meaning: Half of the students scored below 6.5, and the other half scored above 6.5. The Median is not affected by extremely low or high values (outliers).

Variance

Definition: Variance measures the spread of data points around the Mean. A larger value indicates that the data points are more widely dispersed.

Formula:

$\text{Variance} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}^2)$^2
Where:

- $\bar{x}$ is the Mean
- x_i is the score value
- n is the number of students

Example Application (Mean = 6.6):

$\text{Variance} = \frac{1}{10}\left[(3-6.6)^2 + (4-6.6)^2 + \cdots + (10-6.6)^2\right]$ = 44.24
(Detailed calculation can be provided if needed.)

✅
Meaning: A Variance of 4.24 indicates that the scores are not too concentrated, but show a moderate spread around the mean of 6.6.

Standard Deviation

Definition: This is the square root of the Variance, representing the average amount of dispersion of values around the Mean.

Formula:

$\text{Standard Deviation} = \sqrt{\text{Variance}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$

Example Application:

$\text{Standard Deviation} = \sqrt{4.24} \approx 2.06$

✅
Meaning:
→ The majority of students' scores will fall within the range:

6.6
$\pm$ 2.06 $\Rightarrow$ [4.54, 8.66]
This means that most students' scores range from 4.5 to 8.7, indicating a relatively even distribution of data.

Summary Table

Concept	Symbol	Mathematical Formula	Result (Example)	Meaning
Mean	$\bar{x}$	$\frac{1}{n}\sum x_i$	6.6	Average score
Median	—	$x_{\frac{n}{2}}$ (odd n) or $\frac{x_{\frac{n}{2}}+x_{\frac{n}{2}+1}}{2}$ (even n)	6.5	Middle score
Variance	σ^2	$\frac{1}{n}\sum(x_i-\bar{x})^2$	4.24	Data spread level
Standard Deviation	σ	$\sqrt{\frac{1}{n}\sum(x_i-\bar{x})^2}$	2.06	Average deviation level

2.2 Probability Distributions

A probability distribution describes all the possible values a random variable can take and the probability of each of those values occurring. There are two main types of random variables we commonly encounter:

- Discrete Random Variable: This variable can only take countable values (e.g., the number of heads when flipping a coin 5 times, the number of students in a class).
- Continuous Random Variable: This variable can take any value within a continuous range (e.g., a person's height, room temperature, bus waiting time).

Depending on the type of random variable, we will have different probability distribution functions.

Probability Mass Function (PMF)

Concept: The PMF is used for discrete random variables. It gives the probability that a discrete random variable X takes on a specific value x.

Formula: $P(X = x)$ or $f(x)$

Where:

- X is the discrete random variable.
- x is a specific value that X can take.
- $P(X = x)$ is the probability that the variable X takes the value x.

Properties of PMF:

- $0 \leq P(X = x) \leq 1$ for all possible values of x.
- $\sum xP(X = x) = 1$ (the sum of probabilities for all possible values equals 1).

Example:

Suppose you roll a fair six-sided die. The random variable X is the number shown on the die face.

The possible values for X are $1, 2, 3, 4, 5, 6$.

The PMF of X will be:

- $P(X = 1) = 1/6$

- $P(X = 2) = 1/6$
- $P(X = 3) = 1/6$
- $P(X = 4) = 1/6$
- $P(X = 5) = 1/6$
- $P(X = 6) = 1/6$

You can see that the sum of all these probabilities is $1/6 + 1/6 + 1/6 + 1/6 + 1/6 + 1/6 = 6/6 = 1$

Probability Density Function (PDF)

Concept: The PDF is used for continuous random variables. Unlike the PMF, the PDF does not give us the probability of a specific point (because the probability of a single point in a continuous distribution is 0). Instead, the PDF gives us the "density" of probability at a point, and the area under the PDF curve between two points gives us the probability that the random variable falls within that range.

Formula: $f(x)$

Where:

- X is the continuous random variable.
- x is a value that X can take.
- $f(x)$ is the probability density at point x.

Properties of PDF:

- $f(x) \geq 0$ for all values of x.
- $\int_{-\infty}^{\infty} f(x)\,dx = 1$ (the area under the entire PDF curve equals 1).
- $P(a \leq X \leq b)$= $\int_a^b f(x)\,dx$ (the probability that X is within the range [a,b] is the area under the PDF curve from a to b).

Example:

Suppose the height of adult males in a certain country follows a Normal Distribution. This is a typical example of a continuous random variable.

The probability density function for a normal distribution has a somewhat complex formula:

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

Where:

- μ is the mean of the height.
- σ is the standard deviation of the height.
- x is a specific height value.

If you want to calculate the probability that a randomly selected man's height is between $170cm$ and $180cm$, you would calculate the integral of $f(x)$ from 170 to 180:

$P(170 \leq X \leq 180) = \int_{170}^{180} f(x)\,dx$

You don't need to calculate this integral manually; statistical software or programming libraries (like SciPy in Python) can help you do it easily.

Cumulative Distribution Function (CDF)

Concept: The CDF applies to both discrete and continuous random variables. It gives us the probability that the random variable X takes a value less than or equal to a specific value x. In other words, it is the cumulative probability up to that point.

Formula:

- For discrete variables: $F(x) = P(X \leq x) = \sum_{t \leq x} P(X = t)$
- For continuous variables: $F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)\, dt$

Properties of CDF:

- $0 \leq F(x) \leq 1$ for all x.
- $F(-\infty) = 0$.
- $F(\infty) = 1$.
- $F(x)$ is a monotonically non-decreasing function.
- $P(a < X \leq b) = F(b) - F(a)$ (for both discrete and continuous).

Relationship between PMF/PDF and CDF:

- From PMF to CDF (discrete): The CDF is the cumulative sum of the PMF.
- From PDF to CDF (continuous): The CDF is the integral of the PDF.
- From CDF to PMF (discrete): $P(X = x) = F(x) - \lim_{t \to x^-} F(t)$ (if values of x are ordered).
- From CDF to PDF (continuous): The PDF is the derivative of the CDF: $f(x) = dxdF(x)$.

Why are these concepts important for a Data Scientist?

- Understanding Data: Probability distributions help us understand how data is distributed, identifying common values and outliers.
- Modelling: Many statistical and machine learning models (e.g., Linear Regression, Logistic Regression, Naive Bayes) assume that input data follows a certain probability distribution.
- Hypothesis Testing: When you want to draw conclusions about a population based on a sample, you use statistical tests that are based on probability distributions.
- Sampling: Understanding distributions helps you generate data samples that simulate the real world.
- Estimation: Estimating the parameters of a distribution (e.g., mean, standard deviation) is a crucial part of data analysis.

2.3 Covariance and Correlation Coefficient

In Data Analysis and Machine Learning, understanding the relationship between variables is important for building accurate models and interpreting phenomena. The key statistical concepts that help us achieve this are **Covariance** and **Correlation Coefficient.**

Covariance

Covariance is a statistical measure how two variables change together, indicating they move in the same or opposite direction [1]. (linear relationship)

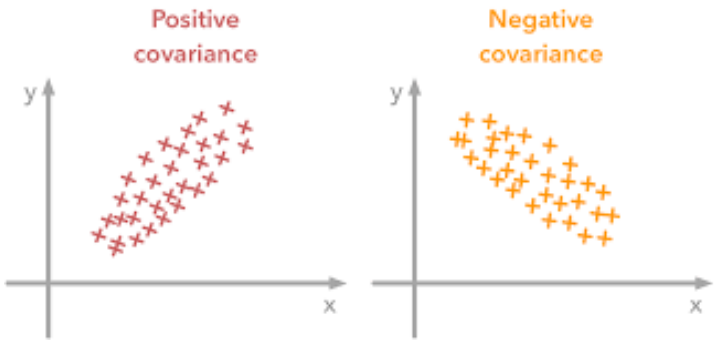


Image 1. Covariance

Let X and Y be two random variables. The covariance between them is defined as:

$$\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)]$$

Where:

- $\mu_X = \mathbb{E}[X]$: expected value (mean) of X
- $\mu_Y = \mathbb{E}[Y]$: expected value (mean) of Y

For a sample of n data points (x_i, y_i) , the sample covariance is estimated as:

$$\mathrm{Cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

Where:

- $\bar{x}$: sample mean of X
- $\bar{y}$: sample mean of Y

Sign of covariance indicates the direction of the relationship:

- $Cov(X, Y) > 0$: X and Y tend to increase together (positive relationship)
- $Cov(X, Y) < 0$: X increases while Y decrease (negative relationship)
- $Cov(X, Y) = 0$: No linear relationship

Covariance Matrix

When working with multiple variables, we use a covariance matrix - a square matrix where each element (i, j) represents the covariance between variable i and variable j.

Given p random variables X_1, X_2,..., X_p , the covariance matrix Σ is:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \cdots & \mathrm{Cov}(X_1, X_p) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}(X_2) & \cdots & \mathrm{Cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_p, X_1) & \mathrm{Cov}(X_p, X_2) & \cdots & \mathrm{Var}(X_p) \end{bmatrix}$$

Where:

- $Var(X_i)$ = $Cov(X_i, X_i)$: variance of variable X_i
- $Cov(X_i, X_j) = Cov(X_j, X_i)$: symmetric property

However, the absolute value of covariance is not easily interpretable, as it depends on ***the units of the variables***.

- We cannot be directly compared across different variable pairs.
- Does not indicate the strength of the relationship (no fixed range).

To overcome these limitations, we use the **Correlation Coefficient (Pearson).**

Pearson Correlation Coefficient

The Pearson Correlation Coefficient is defined as:

$$r_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \cdot \sigma_Y}$$

Where:

- σ_X: standard deviation of X
- σ_Y: standard deviation of Y

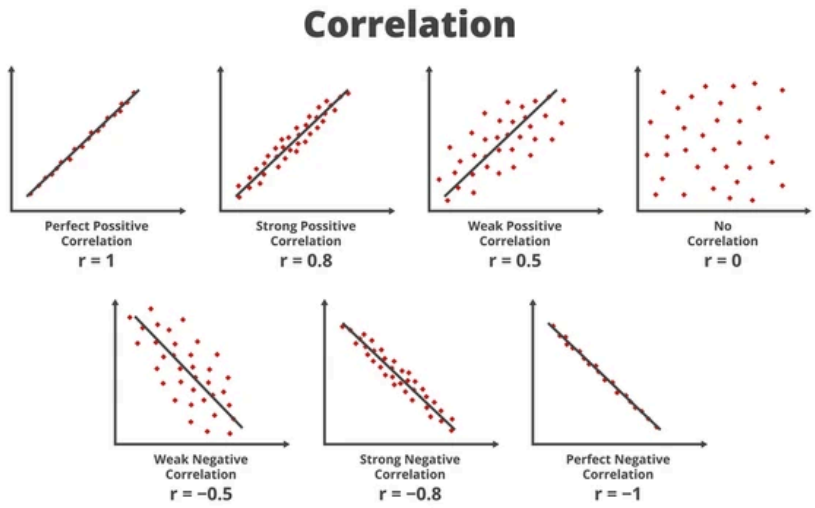


Image 2: Value of Correlation

The Pearson correlation coefficient has the following properties:

- Value in the range: $-1 \leq r_{XY} \leq 1$

- $r_{XY} = 1$: perfect positive linear relationship
- $r_{XY} = -1$: perfect negative linear relationship
- $r_{XY} = 0$: no linear relationship
- The closer to 1 or -1 → the stronger the linear relationship

Correlation standardises the relationship and allows for easier interpretation. However, it have limitations:

- Measures only linear relationships, not ***nonlinear*** **.**
- Sensitive to outliers: heavily influenced or distorted by extreme values in the data
- Does not imply causation : It **does not explain why** variables change together in a linear way.

Covariance and correlation are fundamental yet powerful tools for analysing relationships between variables in statistics and machine learning. Let's go through some real-world data examples to better understand how covariance and correlation work.

3. Experiments

This section will examine the housing price data from a region as follow:

housing_correlation_data.csv

3.1 Data Cleaning with Pandas & NumPy

The journey from raw numbers to valuable insights often begins with a crucial step: **data preprocessing**, or more simply, data cleaning. This process ensures our data is accurate, reliable, and ready for deeper analysis or predictive modeling.

Take housing price from a region as an example, with following key details:

- `Area`: House area (in square meters)
- `Rooms`: Number of rooms
- `Age`: Age of the house (in years)
- `Distance`: Distance to the city center (in kilometers)
- `Price`: House price (target variable in US Dollar per square meters)
- `Crime_Rate`: Crime rate in the area

Let's have a look at the raw data:

Area	Rooms	Age	Distance	Price	Crime_Rate
57.45	50.59	27.00	90.55	41.88	0.61
47.93	47.89	24.62	80.11	37.90	0.69
NaN	40.78	20.30	73.90	39.71	0.43

72.85	61.09	16.77	78.11	25000000	0.38
46.49	33.94	23.49	84.35	27.74	0.28
...	...	...	...	...	NaN
73.69	55.99	24.48	500	53.64	1.09
58.14	42.36	17.32	97.33	44.59	0.24

There are some problem with the data:

- **Missing Values (NaN):** Some entries in `Area` and `Crime_Rate` are simply blank. These gaps can cause problems in our analysis.
- **Outliers:**
  - The `Price` column contains a house with a price of **2,500,000.00**. This value is extremely high compared to others in the dataset and likely represents an outlier, possibly a data entry error or an exceptionally luxurious property.
  - `Distance` has a value of **500.00**, which is an unusually long distance to the city center compared to other entries, also flagging it as an outlier.

Since we're working with numerical data, we should fill in missing values (NaN, empty strings, or null values) with the **median** of each column's existing values.

```
# Normalize all non-numerical values with np.nan first.
df = df.replace(['NA', 'null', ''], np.nan, inplace=True)

# Fill missing numerical values with the median for each column
for column in df.columns:
    if df[column].isnull().any() and df[column].dtype in ['int64', 'float64']:
        median_val = df[column].median()
        df[column].fillna(median_val, inplace=True)
        print(f"Filled missing values in '{column}' with its median: {median_val}")

print("\nMissing values after cleaning (should be 0 for all processed columns):")
print(df.isnull().sum())
print("\nData snapshot after handling missing values:")
print(df.head())
```

```
Missing values after cleaning (should be 0 for all processed columns):
Area          0
Rooms         0
Age           0
Distance      0
Price         0
Crime_Rate    0
dtype: int64

Data snapshot after handling missing values:
    Area  Rooms    Age  Distance  Price  Crime_Rate
0  57.45  50.59  27.00     90.55  41.88        0.61
1  47.93  47.89  24.62     80.11  37.90        0.69
2  59.72  40.78  20.30     73.90  39.71        0.43
3  72.85  61.09  16.77     78.11  53.83        0.38
4  46.49  33.94  23.49     84.35  27.74        0.28
```

With cleaned data, we can conduct experiment to extract the relation between house prices and other features like house area, room numbers or crime rate of the area, ... It ensures that any conclusions drawn or predictions made from our house price data are based on the most robust and trustworthy information available.

3.2 Correlation and it experiment

Take housing price data as an example:

Size: 500 samples × 6 features

- **Variables**:
  - `Area`: House area (in square meters)
  - `Rooms`: Number of rooms
  - `Age`: Age of the house (in years)
  - `Distance`: Distance to the city center (in kilometers)
  - `Price`: House price (target variable)
  - `Crime_Rate`: Crime rate in the area

Using this command to load .csv file

```
def load_from_csv(self, filename="housing_correlation_data.csv"):
    """
    Load data from CSV file for consistent analysis
    """
    import csv
    import os

    if not os.path.exists(filename):
        print(f"[ERROR] File '{filename}' not found!")
        print("   Creating sample data first...")
        self._create_sample_csv(filename)

    try:
        data_dict = {}
        with open(filename, 'r', encoding='utf-8') as csvfile:
            reader = csv.DictReader(csvfile)

            # Initialize columns
            for fieldname in reader.fieldnames:
                data_dict[fieldname] = []

            # Read data
            for row in reader:
                for fieldname in reader.fieldnames:
                    data_dict[fieldname].append(float(row[fieldname]))

        self.data = SimpleDataFrame(data_dict)
```

```
            print(f"[SUCCESS] Data loaded from '{filename}': {self.data.shape}")
            print(f"   • Features: {', '.join(self.data.columns)}")
            return self.data

        except Exception as e:
            print(f"[ERROR] Error loading CSV: {e}")
            return None
```

The sample we will get will look like:

Area	Rooms	Age	Distance	Price	Crime_Rate
57.45	50.59	27.00	90.55	41.88	0.61
47.93	47.89	24.62	80.11	37.90	0.69
59.72	40.78	20.30	73.90	39.71	0.43
72.85	61.09	16.77	78.11	53.83	0.38
46.49	33.94	23.49	84.35	27.74	0.28
46.49	34.75	21.97	81.52	34.31	0.33
73.69	55.99	24.48	84.86	53.64	1.09
61.51	44.89	23.18	91.10	41.77	0.75
42.96	34.61	25.25	88.00	33.84	0.23
58.14	42.36	17.32	97.33	44.59	0.24

Calculate correlation matrix:

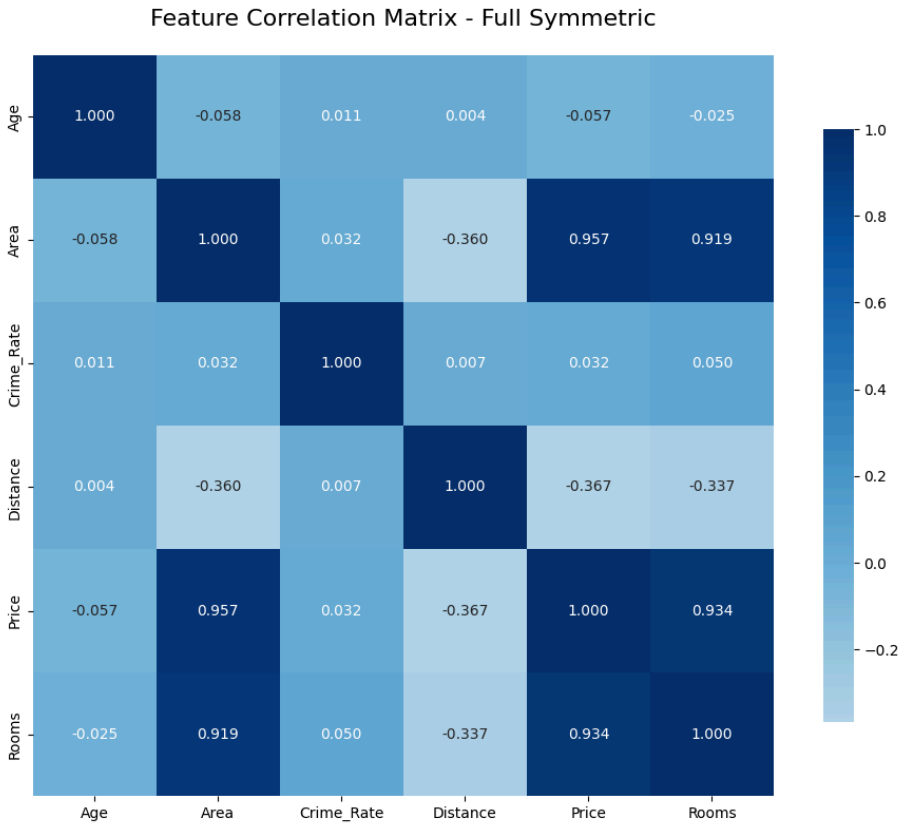
```
def calculate_correlation_matrix(self, method='pearson'):
    """
    Calculate correlation matrix
    """
    if self.data is None:
        raise ValueError("Please load data first!")

    self.correlation_matrix = self.data.corr(method=method)  # ← CALL FUNCTION CORRELATION
    print(f"\nCorrelation matrix ({method}):")
    self.print_correlation_matrix()
    return self.correlation_matrix
```

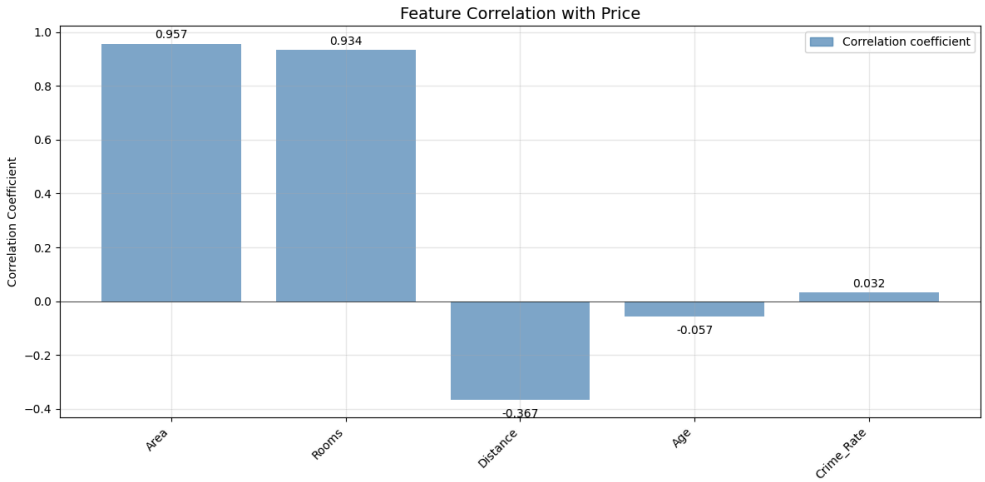
After calculate, we have the matrix:

	Age	Area	Crime Rate	Distance	Price	Rooms
Age	1.000	-0.058	0.011	0.004	-0.057	-0.025
Area	-0,058	1.000	0.032	-0.360	0.957	0.919
Crime Rate	0.011	0.032	1.000	0.007	0.032	0.050
Distance	0.004	-0.360	0.007	1.000	-0.367	-0.337
Price	-0.057	0.957	0.032	-0.367	1.000	0.934
Rooms	-0.025	0.919	0.050	-0.337	0.934	1.000

Representing it with a chart, we have:



From this, we can see that the features that impact house prices the most are:



When building a model to predict price, **Area** and **Rooms** should be prioritised, as they have a strong and positive correlation with the price. Features such as **Crime_Rate** and **Age** can be considered for removal or given less weight, as they have little to no significant impact. **Distance** still holds value, as it shows a moderate and clear negative correlation with price — so it should be retained in the model.

4. Conclusion

In this blog, we explored the critical role of statistics in understanding and preparing data for machine learning. We began with core concepts like **mean**, **median**, **variance**, and **standard deviation**, which help summarise the shape and spread of data distributions. From there, we moved into understanding **covariance** and **correlation**, key tools for identifying relationships between variables — including how strongly features like area and number of rooms relate to house prices.

We then applied these concepts to real-world-style data using Python's panda and numpy libraries. Along the way, we addressed common data challenges, such as **missing values** and **outliers**, and demonstrated how to clean and preprocess data effectively. By calculating and visualising the **correlation matrix**, we gained actionable insights that directly influence model design — such as which features to keep, drop, or transform.

Ultimately, statistical analysis is more than a preliminary step; it's a powerful method for shaping your entire modeling approach. The ability to detect hidden patterns, remove noise, and simplify complexity is what separates a good model from a great one.

Whether you're a beginner or an experienced practitioner, mastering these foundational techniques will enable you to build cleaner, more interpretable, and more accurate models. Keep exploring your data — because the better you understand it, the smarter your models will be.