NAAC Grade 'A"

R. C. PATEL
INSTITUTE OF TECHNOLOGY

An Autonomous Institute

MANTHAN

Department of Computer Science & Engineering (Data Science)

ISSUE YEAR
2023-24

"Information is the resolution of uncertainty."

Claude Shannon
Father of Information theory

Vision-Mission

Institute Vision

To become a leading Institute in Technical education fostering innovation, research, ethical values, and sustainable development for the betterment of society.

Institute Mission

To impart high quality Technical Education through :

- Innovative and Interactive learning process and high quality, internationally recognized instructional programs.
- Fostering a scientific temper among students by the means of a liaison with the Academia, Industries and Government.
- Preparing students from diverse backgrounds to have aptitude for research and spirit of Professionalism.
- To contribute to the nation’s sustainable development.

Vision-Mission

Department Vision

To provide cutting-edge Computer Engineering education in Data Science while instilling socio-moral values.

Department Mission

M1: To deliver state-of-the-art, ICT-enabled teaching and learning to achieve excellence in Data Science education.

M2: To develop professionally competent Data Science Engineers, meeting evolving industrial and societal needs.

M3: To prepare employable professionals with ethical values and a commitment to professional and social responsibility.

Visionary Insights

The Director

Prof. Dr. Jayantrao Bhaurao Patil

At RCPIT, we blend academic rigour with real-world exposure to create future-ready professionals. Emphasizing research, interdisciplinary learning, and soft skills, our hands-on approach empowers students to become ethical leaders, innovators, and entrepreneurs prepared to succeed in a rapidly evolving global landscape.

Visionary Insights

The Head of Department

Prof. Dr. Ujwala Patil

Established in 2020 at RCPIT, Shirpur, the Computer Science & Engineering (Data Science) program trains students in algorithmic thinking and statistical modeling. This interdisciplinary field blends computer science, mathematics, and business analytics. Driven by massive industry demand, it is widely recognized as a highly dynamic and lucrative global career path.

Editor's Message

Prof. A. D. Mairale
Editor (Faculty)

DEAR READERS,

Welcome to the latest edition of our Department Magazine. This magazine is a platform to share knowledge, ideas, and achievements in Computer Science & Engineering (Data Science). It brings together expert articles, career guidance from the Training and Placement Cell, and abstracts of students' publications. We hope these pages inspire curiosity, encourage innovation, and motivate every student to learn continuously. May this magazine strengthen the bond between academia and industry while preparing our students for successful and rewarding careers.

Editorial Board

Mr. Vishal Bhalkar

Mr. Sarthak Patil

Mr. Tejas Chaudhari

Contents

Technical Article

Supervised Learning in Production: Lessons from Industry

Prof. Dr. P. S. Sanjekar

Machine learning is exciting in a classroom. Getting a model to work reliably in the real world is a very different challenge. Industry practitioners learn this the hard way. This article shares the most important lessons from deploying supervised learning models at scale.

Supervised learning is the most widely used branch of machine learning. You provide labelled examples, the algorithm learns a pattern, and it predicts on new data. The concept is simple. The execution at scale is not.

The Training-Production Gap

The biggest surprise for new data scientists is the gap between training accuracy and real-world performance. A model achieving 95% accuracy in testing can fail dramatically after deployment. The reason is data drift: the statistical properties of real-world inputs shift over time.

Customer behaviour changes with seasons. New product categories appear. Market conditions shift. Your model was trained on old data and never saw these new realities. It begins making increasingly poor predictions without any obvious warning.

The solution is continuous monitoring. Every production ML system needs dashboards that track model performance over time. When a metric drops below a threshold, an alert fires and retraining begins automatically.

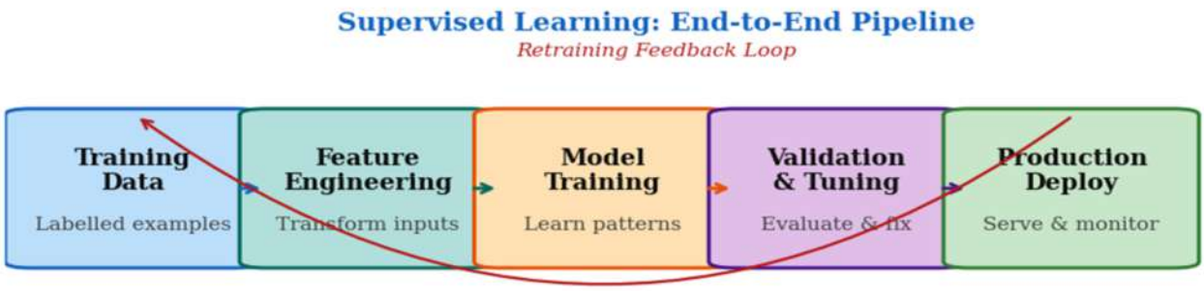


Fig. 1 — Supervised learning pipeline from data to deployment, with a retraining feedback loop

Feature Engineering Is Never Finished

In competitions, you engineer features once and submit. In production, feature engineering is a living process. New data sources become available.

Technical Article

Supervised Learning in Production: Lessons from Industry

Business requirements evolve. Features that mattered two years ago may now be irrelevant or unavailable. Maintain a feature store: a centralised repository that tracks every feature, its definition, and its version history. Tools like Feast and Tecton make this manageable. A well-maintained feature store can reduce the time to retrain and redeploy a model from weeks to hours.

Model Versioning and Rollback

Never deploy a model without version control. If a new version performs poorly, you must roll back to the previous version instantly. MLflow is the industry standard for tracking experiments, storing model artefacts, and managing deployments. A useful best practice is canary deployment. Route 5% of traffic to the new model first. Monitor it carefully. Only when it proves itself do you gradually increase traffic to 100%. This limits the blast radius of any unforeseen failure.

Handling Class Imbalance

Real-world datasets are almost never balanced. A fraud detection dataset may have one fraud case per thousand normal transactions. A naive model that predicts "normal" every time still achieves 99.9% accuracy. That accuracy is completely useless. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) and class-weighted loss functions address this. Always report precision, recall, and F1 score alongside accuracy. These metrics reveal the true story.

Interpretability in High-Stakes Domains

In healthcare, finance, and legal applications, a model decision without an explanation is unacceptable. Doctors will not trust a diagnosis from a black-box system. Tools like SHAP and LIME explain individual predictions in human-understandable terms.

Build interpretability into your deployment practice from the beginning. It is not an optional add-on for auditors. It is a core requirement for gaining trust and adoption in any domain where the stakes are high.

Communication Is a Technical Skill

A production ML system serves business stakeholders, not just data scientists. You must explain model decisions in plain language. Why did the model flag this transaction as fraudulent? What drove the sales forecast upward this quarter?

The most technically brilliant model is worthless if no one trusts or uses it. Invest as much effort in communication and documentation as you invest in model accuracy. Your real-world impact depends on it entirely.

Technical Article

Feature Engineering: The Art Behind the Algorithm

Prof. S. K. Bhandare

When students first encounter machine learning, they focus on algorithms. Which model should I use? Should I try a neural network or a gradient boosted tree? These are natural questions. But experienced practitioners know a deeper truth: the quality of your features matters far more than the choice of algorithm.

Feature engineering is the process of creating informative, relevant, and well-structured input variables for a machine learning model. It sits at the heart of data science practice. It requires both domain knowledge and technical creativity in equal measure.

What Makes a Good Feature?

A good feature is informative: it correlates meaningfully with the target variable. A good feature is independent: it does not duplicate information from other features. A good feature is reliable: it is consistently available at prediction time without leaking future information.

Data leakage is a subtle but critical trap. A feature that uses future data to predict the past produces artificially high training accuracy and fails completely in deployment. Always trace the origin and timing of every feature you engineer with great care.

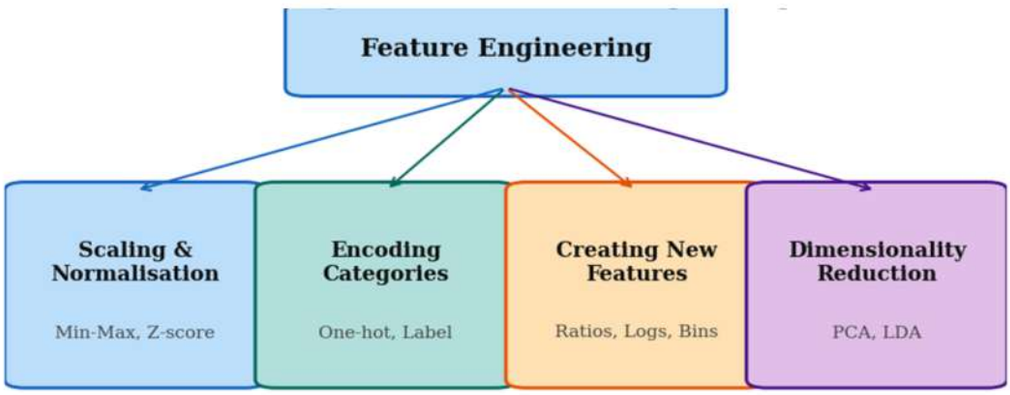


Fig. 2 — The four major families of feature engineering techniques

Numerical Feature Transformations

Raw numbers often need transformation before a model can use them effectively. Log transformation compresses large ranges and makes right-skewed distributions more symmetric. This is helpful for variables like income or population that span several orders of magnitude.

Technical Article

Feature Engineering: The Art Behind the Algorithm

Scaling is equally important. Algorithms like Support Vector Machines and k-Nearest Neighbours are sensitive to the absolute magnitude of features. Min-Max scaling constrains all features to a common range of zero to one. Z-score standardisation shifts features to zero mean and unit standard deviation.

Encoding Categorical Variables

Machine learning algorithms expect numbers, not text. Categorical variables like city names or product categories must be converted. One-hot encoding creates a binary column for each unique category. It works well when the number of categories is small and manageable.

For high-cardinality variables with hundreds of unique values, one-hot encoding becomes impractical. Target encoding replaces each category with the mean of the target variable for that category. Embedding layers in neural networks learn dense representations automatically.

Creating New Features from Existing Ones

Some of the most powerful features are created by combining existing columns intelligently. The ratio of outstanding loan to annual income is more predictive of default than either value alone. Days since last purchase is more useful than a raw purchase timestamp.

Polynomial features capture non-linear relationships. Interaction terms capture combined effects of two variables. Domain knowledge guides these creative combinations. A financial analyst immediately suggests debt-to-income ratio. A meteorologist suggests a temperature-humidity interaction index.

Temporal Features

From a single timestamp you can extract the hour of day, day of week, week of year, month, quarter, and whether a date is a public holiday. These components often reveal powerful patterns that raw timestamps cannot.

Customer support call volume peaks on Monday mornings. Retail sales spike on weekends. Electricity consumption follows daily and seasonal cycles. Extracting temporal components gives your model the ability to learn and reproduce these patterns precisely.

The Iterative Process

Feature engineering is not a one-time task. It is an iterative cycle. Engineer features, train a model, examine which features the model uses most via SHAP values, then create new features informed by what you learn. Repeat until the model performs as required.

Every domain has its own vocabulary of informative features. The data scientist who understands that domain will consistently outperform the one who applies algorithms blindly. Read domain literature. Talk to subject matter experts. Let their knowledge guide every feature you create.

Technical Article

Regression to Classification: A Practical ML Primer

Prof. S. V. Chaudhari

Machine learning solves two broad categories of prediction problems. When the output is a continuous number, such as tomorrow's temperature or next month's revenue, the task is regression. When the output is a category, such as spam or not-spam, the task is classification. This article builds your practical understanding of both, using Python and Scikit-learn throughout.

Scikit-learn is the most widely used machine learning library for structured data. It provides a clean, consistent interface across dozens of algorithms. Once you learn how one algorithm works in Scikit-learn, switching to another requires minimal code changes.

The Machine Learning Workflow

Every ML project follows the same sequence. First, collect and clean your data. Second, split it into training and test sets. Third, select and train a model. Fourth, evaluate performance. Fifth, tune and improve. The test set must never influence training in any way. That separation is sacred.

Linear Regression: Predicting Numbers

Linear regression fits a straight line through data points. It predicts a continuous output as a weighted sum of inputs. In Scikit-learn it takes three lines of code: import LinearRegression, call fit() on training data, then call predict() on test data.

The key metric is Root Mean Squared Error (RMSE). It measures the average distance between predictions and actual values in the original units. Always compare your model's RMSE against a simple baseline such as always predicting the mean. A model that cannot beat the mean is not useful.

Logistic Regression: Classification Despite the Name

Despite its name, logistic regression is a classification algorithm. It predicts the probability that a data point belongs to a particular class. A threshold of 0.5 converts this probability into a class label. It is fast, interpretable, and surprisingly powerful for linearly separable problems.

Logistic regression is the ideal starting point for any classification task. Understand it thoroughly before moving to complex algorithms. If your logistic regression model already achieves 90% accuracy, a random forest that achieves 92% may not justify the added complexity.

Technical Article

Regression to Classification: A Practical ML Primer

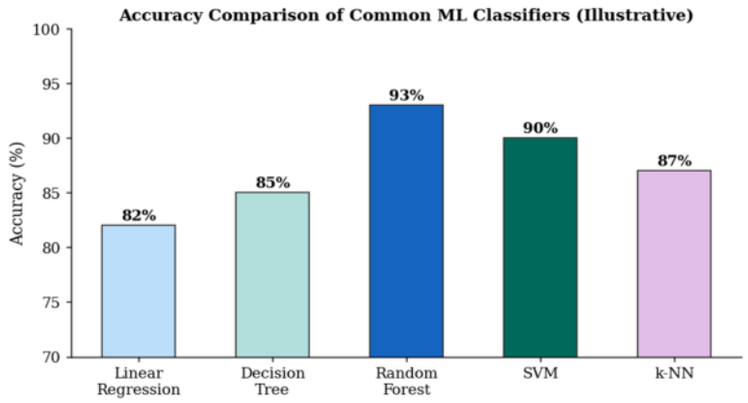


Fig. 3 — Illustrative accuracy comparison of common ML classifiers on a benchmark dataset

Decision Trees and Random Forests

A decision tree makes predictions by asking a sequence of yes-or-no questions about features. It partitions the input space into regions, each associated with a prediction. Decision trees are highly interpretable but prone to overfitting on training data.

A random forest builds many decision trees on random subsets of the data and features, then averages their predictions. This ensemble approach dramatically reduces overfitting. Random forests routinely achieve strong performance with little tuning, making them an excellent first choice for structured tabular data.

The Bias-Variance Tradeoff

Every model makes errors from two sources: bias and variance. High bias means the model is too simple and underfits the data. High variance means the model is too complex and overfits the training data. The art of machine learning is finding the sweet spot between these two.

Decision trees with no depth limit have very high variance. Random forests reduce variance by averaging across many trees. Regularisation adds a penalty for complexity to reduce variance in linear models. Understanding this tradeoff guides every modelling decision you will ever make.

Cross-Validation: A More Honest Evaluation

Evaluating on a single test split can be misleading, especially with small datasets. K-fold cross-validation splits data into k equal folds, trains on k-1 folds, and evaluates on the remaining fold. This repeats k times, rotating the held-out fold each time.

Technical Article

Clustering and Unsupervised Techniques in Student Projects

Prof. S. V. Chaudhari

Not all machine learning tasks involve a labelled target variable. Sometimes the goal is to discover hidden structure within the data itself. This is unsupervised learning, and it is one of the most creative areas of data science. This article explores clustering, the most commonly used unsupervised technique, with examples perfectly suited to undergraduate projects.

Clustering groups data points together based on similarity. Points within the same cluster should resemble each other more than they resemble points in other clusters. No labels are required. The algorithm discovers the groups entirely on its own.

K-Means Clustering

K-means is the most popular clustering algorithm. You specify k, the number of clusters you want to discover. The algorithm then alternates between two steps until convergence: assign each point to the nearest centroid, then move each centroid to the mean of all points assigned to it.

K-means is fast, simple, and works well on compact spherical clusters. Its main limitation is that you must specify k in advance. The elbow method helps: run K-means for several values of k and plot the within-cluster sum of squares. The optimal k sits at the point where the curve bends like an elbow.

A Real Student Project: Customer Segmentation

Imagine you have data on 200 college canteen customers. Each row contains two features: monthly spending and visit frequency. You want to discover distinct customer segments. This is a perfect use case for K-means.

After scaling features to a common range, run K-means with k equal to three. The algorithm identifies three clear segments: occasional visitors who spend little, regular visitors with moderate spending, and high-frequency heavy spenders. Each segment suggests a different canteen strategy and tells a clear business story.

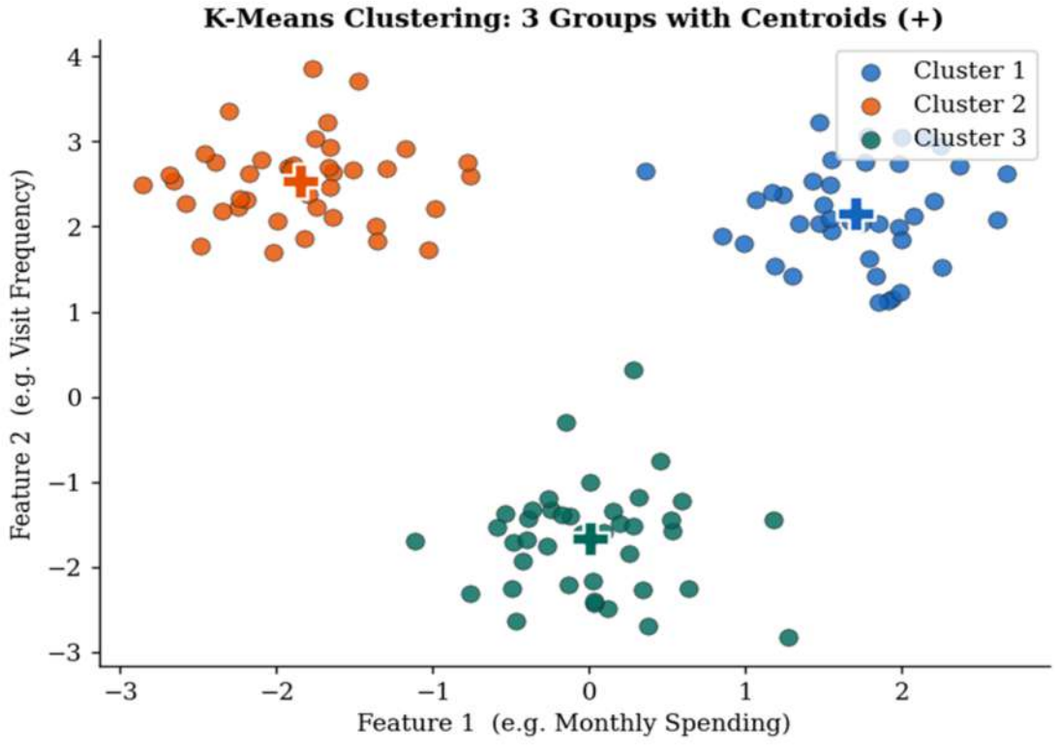
Hierarchical Clustering

Hierarchical clustering builds a tree of clusters called a dendrogram. Agglomerative hierarchical clustering starts with each point as its own cluster. It then repeatedly merges the two closest clusters until only one remains. The dendrogram lets you choose the number of clusters after the fact by cutting the tree at different heights.

Technical Article

Clustering and Unsupervised Techniques in Student Projects

Hierarchical clustering is particularly useful for exploratory analysis where you do not know how many groups to expect. It also handles clusters of irregular shapes, unlike K-means. The trade-off is computational cost: it scales poorly to datasets with hundreds of thousands of data points.



DBSCANDensity-Based Clustering

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It finds dense regions in the data automatically and treats sparse regions as noise or outliers. It is effective when clusters have irregular shapes and when outlier detection is also a goal.

DBSCAN requires two parameters: epsilon, the neighbourhood radius, and min_samples, the minimum number of points to form a dense region. It does not require specifying the number of clusters in advance. Choosing epsilon and min_samples thoughtfully is the main skill in using DBSCAN well.

Technical Article

Clustering and Unsupervised Techniques in Student Projects

Dimensionality Reduction with PCA

Principal Component Analysis reduces the number of dimensions while retaining as much variance as possible. The first principal component is the direction of greatest variance in the data. Each subsequent component is perpendicular to all previous ones and captures the next highest variance available.

In student projects, use PCA to visualise high-dimensional datasets in two dimensions. After applying PCA, plot the first two components and colour-code the points by their cluster label. This visualisation reveals the structure that the clustering algorithm discovered in a form that any reader can immediately understand and interpret.

Unsupervised learning is genuinely exploratory. There is no single correct answer. Different algorithms and parameter choices reveal different aspects of the same data. Embrace this ambiguity. The most interesting insights in data science often come from discovering unexpected structure in data that was not labelled to tell you what to find.

Career Horizon

How to Land Your First Data Science Internship

Prof. T. R. Girase

An internship is the most valuable single asset on an engineering student's resume. It signals real-world experience, professional conduct, and the ability to deliver results under pressure. For data science students, a well-chosen internship can unlock full-time roles, research collaborations, and professional networks that last throughout your entire career.

This article gives you a concrete, actionable roadmap to landing your first data science internship. It is based on what actually works for students from engineering colleges, not generic advice from career books.

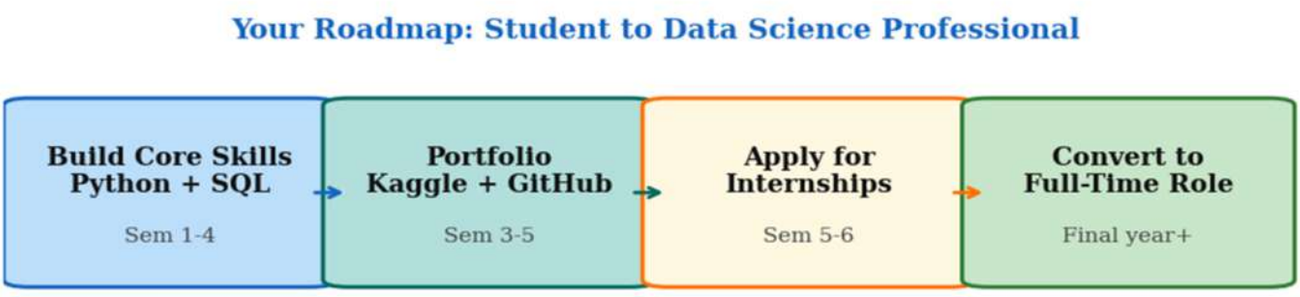


Fig. 5 — Your step-by-step roadmap: from student to data science professional

Build the Foundation First

Internship applications without a foundation of skills go nowhere. Before applying anywhere, ensure you can confidently answer yes to three questions. Can I write Python code to load, clean, and visualise a dataset? Can I build and evaluate a machine learning model using Scikit-learn? Can I explain what I did and why to a non-technical person?

These are the table stakes. Interviewers at strong companies test these skills directly in the interview. Online courses from Kaggle Learn, Coursera, and fast.ai provide structured pathways. Combine every course with project work. Reading without coding is not learning in data science.

The GitHub Portfolio

Every data science recruiter looks at your GitHub profile before an interview. If it is empty, that is a significant disadvantage. Your GitHub should contain at least three complete, well-documented projects.

Technical Article

How to Land Your First Data Science Internship

Each project needs a clear README explaining the problem, your approach, and what you found.

Strong projects solve real problems with real data. Predicting air quality in your city using government sensor data is far more impressive than the tenth student project on the Titanic dataset. Use data that genuinely interests you. Your authentic curiosity will show clearly in the quality and depth of your analysis.

Kaggle: Competing and Learning

Kaggle is the world's largest platform for data science competitions. Even if you never win, participating builds skills that courses cannot teach. You learn to read other people's notebooks, discover techniques you had not considered, and practise the discipline of iterative improvement under a leaderboard constraint.

Begin with beginner competitions. The Titanic survival prediction is the standard starting point. Then move to active competitions and focus on understanding the data deeply rather than chasing the leaderboard. Your completed Kaggle notebooks are a portfolio in themselves and visible to every recruiter.

Targeting the Right Companies

Not all internships are equal. A data science internship at a company that only uses Excel for analysis will teach you very little. Target companies with a genuine data science practice: analytics firms, fintech startups, e-commerce companies, and healthtech organisations.

In India, Bangalore, Mumbai, Hyderabad, and Pune have the densest concentrations of data science roles. Remote internships with international companies are also increasingly available and should not be overlooked. Check company job boards directly alongside platforms like LinkedIn, Internshala, and AngelList.

Acing the Data Science Interview

Data science interviews typically have three components. First, a take-home assignment where you analyse a real dataset and present findings. Second, a technical interview testing Python, SQL, and ML concepts. Third, a case interview where you solve an open-ended business problem with data.

Technical Article

How to Land Your First Data Science Internship

Practise SQL daily on HackerRank or LeetCode. SQL is tested at nearly every data company, yet many students neglect it entirely. For the case interview, practise structuring your thinking aloud. Interviewers care about your reasoning process as much as your final answer. Articulating your thinking clearly is a skill worth practising deliberately.

Soft Skills Win Offers

At the final stage, most candidates have comparable technical skills. What differentiates them is communication, attitude, and cultural fit. Be genuinely curious about the company's data challenges. Ask thoughtful questions. Send a follow-up note of thanks after the interview.

The first internship is the hardest to get. Every subsequent one becomes easier. Start early, build consistently, and apply to many companies at once. One yes is all you need. That first yes changes the trajectory of everything that follows.

Selected Student Articles

SONAR Rock vs Mine Prediction

Abstract
In marine operations, sonar devices are essential, especially for finding submerged objects like rocks and mines. Ensuring maritime safety and security requires the ability to distinguish between these things with accuracy. We provide a thorough comparison of machine learning algorithms in this research study to help determine if sonar returns are indicative of rocks or mines. Using a dataset of sonar sound characteristics, we assess several supervised learning algorithms: random forests, k-nearest neighbours, support vector machines, and deep neural networks. We examine the models' performance in terms of F1-score, recall, accuracy, and precision in classification. We also investigate how feature selection strategies and hyperparameter adjustments affect the performance of the model. By means of comprehensive testing and analysis, we offer insights into the effectiveness of different machine learning approaches for sonar-based object classification, ultimately contributing to enhanced maritime security and underwater navigation systems.

Keywords – SONAR, Underwater Object Classification, Maritime Security, Rock Detection, Mines Detection, Supervised machine learning, Feature Selection, Hyperparameter Tuning.

Shruti Pathak, Vaibhavi Raul, Pushpak Desale, Deep Bhandari, and Dr. Priti Sanjekar

Selected Student Articles

Sentiment Analysis using ktrain

Abstract

This study uses the ktrain library, a thin wrapper around TensorFlow Keras, to provide a thorough method of sentiment analysis. The paper investigates how ktrain can streamline the creation, training, and implementation of machine learning models for sentiment analysis applications. The research shows how efficient and effective ktrain is at achieving high accuracy with minimal computational resources by using pre-trained models and fine-tuning them on a sentiment-labeled dataset. The methodology offers a clear workflow for sentiment analysis applications, comprising steps for data preprocessing, model selection, training, evaluation, and deployment. The findings show that ktrain greatly lowers the time and complexity needed for sentiment analysis, making it usable by practitioners of all skill levels. This paper contributes to the field by offering a practical guide and highlighting the potential of ktrain in natural language processing tasks.

Keywords – ktrain, sentiment, computational, preprocessing, deployment.

Vedant Thakur, Shivam Yogi, Dipak Mali, Shruti Nhavalde, and Prof. S. K. Bhandare.