WOMEN WHO CODE
/data-science

Build a Data Science Project From Scratch - SESSION 5

JULY 7
10 PM ET

Siddhi Purohit
Upcoming Data Analyst
at J.P.Morgan

womenwhocode.com/datascience/events

Session Agenda

- Model selection importance and process
- Importance of understanding the underlying statistics of how algorithms work
- Common advanced models
- Complexity of model
- Tuning and optimizing models
- Selecting our models
- Session use case & colab work

Significance of Model Selection

Plays a key role in the success of the project

- Well suited models impact the performance and accuracy of predictions.
- Ensures the model's assumptions align with the characteristics of the data, the solution and the use case.
    - What are the real world consequences of results?
    - Poorly chosen models may underfit or overfit the data.
    - Overfitting occurs when a model performs well on the training data but poorly on new, unseen data
    - Underfitting refers to a model that fails to capture the underlying patterns in the training data and performs poorly both on the training and new data.

Model Selection Process

- Problem definition and data understanding
- Feature selection and data preprocessing
- Algorithm selection based on problem characteristics
- Hyperparameter tuning and model training
- Evaluation and comparison of model performance

Studying Statistics and Algorithm Matter

- Improved understanding of underlying assumptions and limitations of models.
- Enhanced ability to choose appropriate evaluation metrics.
- Skill in selecting the right algorithm and fine-tuning hyperparameters.
- Increased interpretability and trust in model results.
- Better decision-making and avoidance of common pitfalls.

Advanced ML Models

Machine Learning Model	Supervised/ Unsupervised	Use Cases and Features	Pros	Cons
Support Vector Machines (SVM)	Supervised	Effective for linear and non-linear classification with clear separation between classes.	Works well in high-dimensional spaces.	Can be computationally expensive for large datasets.
Random Forest	Supervised	Robust, handles complex relationships, and works well with high-dimensional data.	Reduces overfitting, provides feature importance.	Can be prone to overfitting noisy data.
Gradient Boosting algorithms (e.g., XGBoost, LightGBM)	Supervised	Improves predictive accuracy, handles complex features and relationships.	Can handle complex non-linear relationships.	Requires careful tuning to avoid overfitting.

Advanced ML Models

Machine Learning Model	Supervised/ Unsupervised	Use Cases and Features	Pros	Cons
Linear Regression	Supervised	Predicting numeric values with a linear relationship between features and the target variable.	Simple interpretation, fast training.	Assumes a linear relationship between variables.
Decision Trees	Supervised	Versatile, handles both numeric and categorical features, captures non-linear relationships.	Easy to interpret and visualize.	Prone to overfitting without proper regularization.
Neural Networks	Supervised	Deep learning models like MLPs, effective for regression with large datasets and complex patterns.	Can capture complex patterns and relationships.	Requires large amounts of data, computationally intensive training.

Advanced ML Models

Machine Learning Model	Supervised/ Unsupervised	Use Cases and Features	Pros	Cons
Collaborative Filtering	Unsupervised	Personalized recommendations based on user-item interactions and similarity between users/items.	Doesn't require explicit feature engineering.	Cold start problem, sparsity of data.
Matrix Factorization	Unsupervised	Decomposes the user-item interaction matrix, captures latent preferences, and makes recommendations.	Handles sparse data, captures latent features.	Cold start problem, doesn't capture complex relationships.
Deep Learning Models	Supervised/Unsupervised	Recurrent Neural Networks (RNNs) and Transformer models for capturing complex patterns in user behavior and item features.	Can capture complex temporal dependencies.	Requires large amounts of data, computationally intensive training.

Advanced ML Models

Machine Learning Model	Supervised/ Unsupervised	Use Cases and Features	Pros	Cons
K-Means	Unsupervised	Partitioning data into K clusters based on similarity. Suitable when the number of clusters is known or estimated.	Simple and fast algorithm.	Sensitive to initial cluster centroids, may converge to local optima.
Hierarchical Clustering	Unsupervised	Builds a hierarchy of clusters, capturing hierarchical structures in the data.	No need to pre-specify the number of clusters.	Computationally expensive for large datasets.
Density-Based Spatial Clustering (DBSCAN)	Unsupervised	Identifies clusters based on density, suitable for arbitrary-shaped clusters and handling noise.	Can discover clusters of different shapes and sizes.	Sensitive to the choice of distance metric and density parameters.

Advanced ML Models

Large Language Models (LLMs)	Supervised/ Unsupervised	Use Cases and Features	Pros	Cons
BERT (Bidirectional Encoder Representations from Transformers)	Unsupervised	Language understanding, text classification, named entity recognition, question answering, text generation.	Captures contextual information, pre-trained on large corpus.	Computationally intensive, requires large amounts of training data.
GPT (Generative Pre-trained Transformer)	Unsupervised	Language generation, text completion, text summarization, dialogue systems.	Generates coherent and human-like text, captures long-range dependencies.	Computationally intensive, may generate incorrect or nonsensical text.
GPT-3 (Generative Pre-trained Transformer 3)	Unsupervised	Language translation, sentiment analysis, chatbots, virtual assistants.	High language understanding, supports various language-related tasks.	Resource-intensive, limited availability, potential ethical concerns.
T5 (Text-to-Text Transfer Transformer)	Unsupervised	Multitask learning, text summarization, language translation, document classification.	Unified framework for multiple NLP tasks, supports transfer learning.	Requires large-scale training data, computationally demanding.
XLNet (eXtreme Language Understanding Network)	Unsupervised	Language understanding, sentiment analysis, text classification, question answering.	Overcomes limitations of traditional left-to-right pre-training.	High memory requirements, complex architecture.

Complexity of a Model

How complex should our ML models be?

- More complex generally performs better on training data than simpler ones
- Complex models offer more parameters that are adjustable to get a better fit on the training result
- Too complex can end up overfitting to generate random effects present only in specific datasets and not the unseen test data

Model Specific Techniques

Different algorithms may have specific optimization techniques:

- **Decision Trees:** Pruning, adjusting tree depth, or limiting leaf nodes.
- **Neural Networks:** Adding dropout layers, adjusting learning rate, or changing network architecture.
- **Support Vector Machines:** Tuning the kernel function, C parameter, or gamma value.

Model Tuning - Hyperparameters

Crucial to improve performance and generalization ability of models.

- **Grid Search:** Exhaustively search a predefined hyperparameter grid to find the best combination.
- **Random Search:** Randomly sample from the hyperparameter space to explore a wide range of possibilities.
- **Bayesian Optimization:** Utilize probabilistic models to find promising hyperparameters efficiently.

Model Tuning - Cross Validation

Assess the performance and generalization ability by evaluating it on multiple subsets of the data.

- A more reliable estimate of how well the model will perform on unseen data compared to a single train-test split.
- **K-Fold Cross-Validation:** data is divided into K equally-sized folds. The model is trained and evaluated K times, with each fold serving as the test set once and the remaining folds as the training set.
- **Leave-One-Out Cross-Validation (LOOCV):** Each observation is treated as a separate fold. The model is trained and evaluated using N iterations, where N is the number of observations in the dataset.
- **Stratified Cross-Validation:** Used for imbalanced datasets, it ensures that the class distribution is preserved in each fold, reducing bias in the evaluation.

Cross Validation - Pros

- Better Performance Estimate: Cross-validation provides a more reliable estimate of model performance, reducing the impact of randomness in a single train-test split.

- Model Selection: Cross-validation helps in comparing and selecting the best model among different algorithms or hyperparameter configurations.

- Robustness Check: It helps assess the stability and consistency of the model by evaluating it on different subsets of the data.

Cross Validation - Cons

- Computationally Intensive: Cross-validation requires training and evaluating the model multiple times, which can be time-consuming, especially for large datasets or complex models.

- Information Leakage: if information from the validation/test set leaks into the training set, this can lead to overly optimistic performance estimates or thinking our model is better than it actually is. It might perform well on our test data (because it has indirectly seen it during training), but it may not perform as well on new, unseen data. This can happen if we perform feature selection or preprocessing steps on the entire dataset before splitting it into training and test sets.

Cross-Validation Variants

- **Stratified K-Fold:** Preserves class distribution in each fold, suitable for imbalanced datasets.

- **Shuffle-Split:** Randomly splits the data into train-test sets multiple times, allowing control over the train-test split size and repetition.

- **Time Series Cross-Validation:** Handles temporal dependencies by preserving the chronological order of data during cross-validation.

Model Optimization

1. Model Selection
    - Compare multiple models with different algorithms to identify the best-performing one.
    - Consider factors such as model complexity, interpretability, computational requirements, and suitability for the problem.
2. Feature Selection and Engineering:
    - Identify relevant features and remove irrelevant or redundant ones.
    - Create new features that capture important patterns or interactions in the data.

Model Optimization - Ensemble Methods

- Combine multiple models to create a stronger and more robust ensemble model.

- The goal is to improve prediction accuracy, handle complex relationships, reduce overfitting, and enhance generalization.

- Techniques include bagging (e.g., Random Forest), boosting (e.g., Gradient Boosting), and stacking.

Ensemble Methods

Bagging (Bootstrap Aggregating)

- combines multiple models trained on different subsets of the data, obtained through bootstrapping.
- Each base model is trained independently, and the final prediction is determined through aggregation, such as majority voting for classification or averaging for regression.
- ***Random Forest is a popular bagging ensemble method that combines multiple decision trees.***

Boosting

- builds an ensemble by sequentially training multiple weak models, where each subsequent model focuses on the samples that previous models struggled to predict correctly.
- assigns weights to observations, emphasizing the importance of harder-to-predict instances.
- ***Gradient Boosting algorithms (e.g., XGBoost, LightGBM) and AdaBoost***

Stacking

- combines predictions from multiple base models by training a meta-model, also known as a blender or a meta-learner, on the predictions of the base models.
- The meta-model learns to make predictions based on the outputs of the base models, effectively capturing their collective knowledge.
- allows for the exploitation of diverse model behaviors and can lead to improved performance.

Ensemble Pros

- Improved Accuracy: Ensemble methods leverage the collective knowledge of multiple models, reducing bias and improving prediction accuracy.
- Handling Complexity: Ensemble methods can handle complex relationships and capture nonlinear patterns that may be challenging for individual models.
- Reducing Overfitting: Combining models reduces the risk of overfitting by reducing variance and balancing model predictions.
- Feature Importance: Ensemble methods can provide insights into feature importance by aggregating information from multiple models.

Ensemble Considerations

- **Model Diversity:** Ensemble models benefit from diverse base models, capturing different aspects of the data and reducing errors caused by individual models' weaknesses.
- **Computational Complexity:** Ensemble methods are generally more computationally intensive and may require more resources than individual models.
- **Hyperparameter Tuning:** Ensemble methods have additional hyperparameters that need to be optimized to achieve optimal performance.

Model Optimization - Regularization

- Used to prevent overfitting
- Works by adding a penalty to the loss function, which discourages the model from learning overly complex patterns in the training data.
- L1 Regularization (Lasso): adds a penalty equal to the absolute value of the magnitude of the coefficients. This can be useful when we have a high-dimensional dataset and we suspect that some features might not be relevant.
- L2 Regularization (Ridge): adds a penalty equal to the square of the magnitude of the coefficients. This is useful when we believe that all features are relevant.
- Control model complexity and penalize large coefficient values.

Model Evaluation and Validation

- Use appropriate evaluation metrics based on the problem type (classification, regression, etc.).

- Validate the model on an independent test set to ensure generalization.

Constant Retraining is Important!

- **Change in Business Scope:** Ensures they align with the current business objectives and adapt to new requirements.
- **New Data Influx:** As new data becomes available, retraining models allows them to incorporate the latest information.
- **Data Drift:** occurs when the statistical properties of the input data change, affecting the model's performance. Regular retraining helps models adapt to these changes and maintain their accuracy.
- **External Factors:** such as changes in customer behavior, market conditions, regulations, or competitive landscape, can impact the relevance and performance of models.
- **Model Maintenance:** require ongoing maintenance to address issues like model degradation, concept drift, or the emergence of new biases.
- **Performance Monitoring:** allows for continuous performance monitoring and evaluation of models.

Our Model Selection

- **Logistic Regression:** easy to interpret ,serves as a good baseline
- **Decision Trees:** capture non-linear relationships and interactions between features. Quite interpretable.
- **Random Forests:** ensemble of decision trees. More robust and less prone to overfitting than a single decision tree. A bit less interpretable.
- **Gradient Boosting Machines (like XGBoost or LightGBM):** some of the best performing models for tabular data. They are more complex and require careful tuning.

Join us on Slack to ask questions and keep the discussion going!

Use the channel:

#build-a-ds-project

Let's Build Our Model!

At the end of the session, we will ask for your feedback on the series. Please be sure to share your thoughts with us via the link provided in chat before you log off.