https://sites.google.com/view/aiml-deepthought/home

Classification Metrics: A Comprehensive Guide

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, log_loss, roc_auc_score
from sklearn.metrics import recall_score, f1_score, precision_score, matthews_corrcoef, balanced_accuracy_score
from sklearn.linear_model import LogisticRegression, RidgeClassifierCV, Perceptron, SGDClassifier, RidgeClassif
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifie
from sklearn.neighbors import NearestCentroid, KNeighborsClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import BernoulliNB, GaussianNB
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
from xgboost import XGBClassifier
import lightgbm as lgb
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_curv
from sklearn.model_selection import train_test_split, cross_val_score
import category_encoders as ce

import warnings
warnings.filterwarnings('ignore')
```

In [13]:
```
# Load and preprocess the dataset
titanic = pd.read_csv('titanic.csv')
titanic = titanic.drop(['PassengerId', 'Name', 'Cabin', 'Ticket'], axis=1)
titanic['Age'] = titanic['Age'].fillna(titanic['Age'].mean())
titanic['Embarked'] = titanic['Embarked'].fillna(titanic['Embarked'].mode()[0])

X = titanic.drop(['Survived'], axis=1)
y = titanic['Survived']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=21)

# Encode categorical variables
encoder = ce.OrdinalEncoder(['Sex', 'Embarked'])
X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)

# Define models
models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(criterion='entropy', n_estimators=100),
    'LightGBM': lgb.LGBMClassifier(),
    'Ridge Classifier CV': RidgeClassifierCV(),
    'XGBoost': XGBClassifier(),
    'Nearest Centroid': NearestCentroid(),
    'Quadratic Discriminant Analysis': QuadraticDiscriminantAnalysis(),
    'Calibrated Classifier CV': CalibratedClassifierCV(),
    'Bernoulli NB': BernoulliNB(),
    'Bagging Classifier': BaggingClassifier(),
    'SVC': SVC(),
    'Linear SVC': LinearSVC(),
    'KNeighbors Classifier': KNeighborsClassifier(),
    'Gaussian NB': GaussianNB(),
    'Perceptron': Perceptron(),
    'SGD Classifier': SGDClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'MLP Classifier': MLPClassifier(),
    'Extra Trees': ExtraTreesClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Nu SVC': NuSVC(),
    'Gaussian Process': GaussianProcessClassifier(kernel=RBF()),
    'Ridge Classifier': RidgeClassifier(),
    'Passive Aggressive': PassiveAggressiveClassifier(),
    'Hist Gradient Boosting': HistGradientBoostingClassifier()
}

# Train and predict
results = []
for name, model in models.items():
```

```
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Calculate metrics
    CM = confusion_matrix(y_test, y_pred)
    TN, FP, FN, TP = CM.ravel()

    specificity = TN / (TN + FP)
    loss_log = log_loss(y_test, y_pred)
    acc = accuracy_score(y_test, y_pred)
    balanced_acc = balanced_accuracy_score(y_test, y_pred)
    roc = roc_auc_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    mathew = matthews_corrcoef(y_test, y_pred)

    results.append([name, acc, balanced_acc, prec, rec, specificity, f1, roc, loss_log, mathew])

model_results = pd.DataFrame(results, columns=['Model', 'Accuracy', 'Balanced Accuracy', 'Precision', 'Sensitiv


model_results = model_results.sort_values('Accuracy', ascending=False).reset_index(drop=True)
```

Classification Metrics: A Comprehensive Guide

When evaluating the performance of a classification model, several metrics are commonly used. Here's a breakdown of each metric, along with its importance:.

Accuracy

- Definition: The ratio of correct predictions to the total number of predictions.
- Importance: A general measure of how well the model performs overall. However, it can be misleading in imbalanced datasets.

Balanced Accuracy

- Definition: The average of sensitivity and specificity, providing a more balanced view in imbalanced datasets.
- Importance: Useful when dealing with imbalanced classes, as it accounts for both positive and negative predictions.

Precision

- Definition: The ratio of true positive predictions to the total number of positive predictions.
- Importance: Measures how many of the positive predictions made by the model were actually correct.

Sensitivity (Recall)

- Definition: The ratio of true positive predictions to the total number of actual positive instances.
- Importance: Measures how well the model can identify positive instances.

Specificity

- Definition: The ratio of true negative predictions to the total number of actual negative instances.
- Importance: Measures how well the model can identify negative instances.

F1 Score

- Definition: The harmonic mean of precision and recall.
- Importance: A balanced metric that considers both precision and recall, making it useful when both are important.

ROC AUC (Receiver Operating Characteristic Area Under the Curve)

- Definition: The area under the ROC curve, which plots the true positive rate (sensitivity) against the false positive rate.
- Importance: A measure of the model's ability to discriminate between positive and negative classes, regardless of the classification threshold.

Log Loss (Cross-Entropy Loss)

- Definition: A measure of the model's predicted probability distribution compared to the actual labels.
- Importance: Often used in machine learning algorithms, especially those that output probabilities. Lower log loss indicates better performance.

Matthews Correlation Coefficient (MCC)

- Definition: A correlation coefficient between predicted and actual labels, considering true positives, true negatives, false positives, and false negatives.
- Importance: A more balanced metric than accuracy, especially for imbalanced datasets. It ranges from -1 (worst) to 1 (best).

Choosing the Right Metrics:

The choice of metrics depends on the specific problem and the relative importance of different aspects of the model's performance. For example:

- Imbalanced datasets: Balanced accuracy, F1 score, MCC, and ROC AUC are often preferred.
- When both precision and recall are important: F1 score is a good choice.
- When probability predictions are important: Log loss is useful.

In many cases, it's helpful to consider multiple metrics to get a comprehensive understanding of the model's performance.

In [7]:
```
model_results
```

Out[7]:

	Model	Accuracy	Balanced Accuracy	Precision	Sensitivity	Specificity	F1 Score	ROC AUC	Log Loss	Matthews Corrcoef
0	Random Forest	0.832402	0.817246	0.843750	0.729730	0.904762	0.782609	0.817246	6.040836	0.651925
1	XGBoost	0.832402	0.821236	0.823529	0.756757	0.885714	0.788732	0.821236	6.040836	0.651851
2	Extra Trees	0.832402	0.823230	0.814286	0.770270	0.876190	0.791667	0.823230	6.040836	0.652365
3	LightGBM	0.821229	0.807722	0.818182	0.729730	0.885714	0.771429	0.807722	6.443558	0.628185
4	Gaussian NB	0.815642	0.802960	0.805970	0.729730	0.876190	0.765957	0.802960	6.644919	0.616566
5	Hist Gradient Boosting	0.810056	0.798198	0.794118	0.729730	0.866667	0.760563	0.798198	6.846281	0.605103
6	Ridge Classifier	0.810056	0.794208	0.812500	0.702703	0.885714	0.753623	0.794208	6.846281	0.604584
7	Logistic Regression	0.810056	0.794208	0.812500	0.702703	0.885714	0.753623	0.794208	6.846281	0.604584
8	Ridge Classifier CV	0.810056	0.794208	0.812500	0.702703	0.885714	0.753623	0.794208	6.846281	0.604584
9	MLP Classifier	0.804469	0.795431	0.774648	0.743243	0.847619	0.758621	0.795431	7.047642	0.594779
10	Quadratic Discriminant Analysis	0.804469	0.787452	0.809524	0.689189	0.885714	0.744526	0.787452	7.047642	0.592797
11	Calibrated Classifier CV	0.798883	0.780695	0.806452	0.675676	0.885714	0.735294	0.780695	7.249003	0.581014
12	Bagging Classifier	0.798883	0.782690	0.796875	0.689189	0.876190	0.739130	0.782690	7.249003	0.580914
13	AdaBoost	0.798883	0.788674	0.771429	0.729730	0.847619	0.750000	0.788674	7.249003	0.582621
14	Nu SVC	0.798883	0.788674	0.771429	0.729730	0.847619	0.750000	0.788674	7.249003	0.582621
15	Perceptron	0.787709	0.783140	0.736842	0.756757	0.809524	0.746667	0.783140	7.651725	0.564179
16	Linear SVC	0.770950	0.776834	0.689655	0.810811	0.742857	0.745342	0.776834	8.255809	0.545515
17	Decision Tree	0.765363	0.752124	0.735294	0.675676	0.828571	0.704225	0.752124	8.457170	0.511609
18	SGD Classifier	0.709497	0.736422	0.600000	0.891892	0.580952	0.717391	0.736422	10.470782	0.478418
19	Gaussian Process	0.703911	0.663835	0.744186	0.432432	0.895238	0.547009	0.663835	10.672143	0.377698
20	KNeighbors Classifier	0.698324	0.671042	0.678571	0.513514	0.828571	0.584615	0.671042	10.873504	0.363327
21	Passive Aggressive	0.692737	0.642342	0.787879	0.351351	0.933333	0.485981	0.642342	11.074866	0.361527
22	SVC	0.664804	0.616538	0.694444	0.337838	0.895238	0.454545	0.616538	12.081672	0.286344
23	Nearest Centroid	0.648045	0.612227	0.612245	0.405405	0.819048	0.487805	0.612227	12.685755	0.247894
24	Bernoulli NB	0.558659	0.506113	0.428571	0.202703	0.809524	0.275229	0.506113	15.907534	0.015181

Relationships Between Classification Metrics and Choosing the Right Ones for Generalized Results

Understanding the relationships between classification metrics and choosing the appropriate ones for generalized results is crucial in machine learning. Let's delve into these aspects:

Relationships Between Metrics

- **Complementary:** Some metrics provide complementary information. For instance, precision and recall can be combined to understand the trade-off between identifying positive instances correctly (precision) and identifying all positive instances (recall).
- **Derived:** Some metrics are derived from others. The F1 score, for example, is the harmonic mean of precision and recall.
- **Trade-offs:** Often, there is a trade-off between different metrics. For instance, increasing precision might decrease recall, and vice versa.

Choosing Metrics for Generalized Results

When selecting metrics for generalized results, consider the following factors:

1. Dataset Characteristics:

- **Imbalanced Classes:** If the dataset is imbalanced (one class has significantly more instances than the other), metrics like balanced accuracy, F1 score, and MCC are more appropriate.
- **Noise:** If the dataset contains noise or outliers, metrics that are less sensitive to individual errors, such as ROC AUC, can be helpful.

2. Problem Domain:

- **False Positives vs. False Negatives:** Consider the consequences of different types of errors. If false positives are more costly, focus on precision. If false negatives are more costly, focus on recall.
- **Probability Predictions:** If the model outputs probabilities, metrics like log loss can be used to evaluate the quality of the probability distribution.

3. Evaluation Goals:

- **Overall Performance:** Accuracy can be a good starting point, but be aware of its limitations in imbalanced datasets.
- **Class-Specific Performance:** If you need to evaluate performance for specific classes, calculate metrics for each class.
- **Trade-offs:** If there is a trade-off between different metrics, consider using a composite metric like the F1 score or creating a weighted average based on the relative importance of each metric.

4. Interpretability:

- Choose metrics that are easy to interpret and communicate to stakeholders.

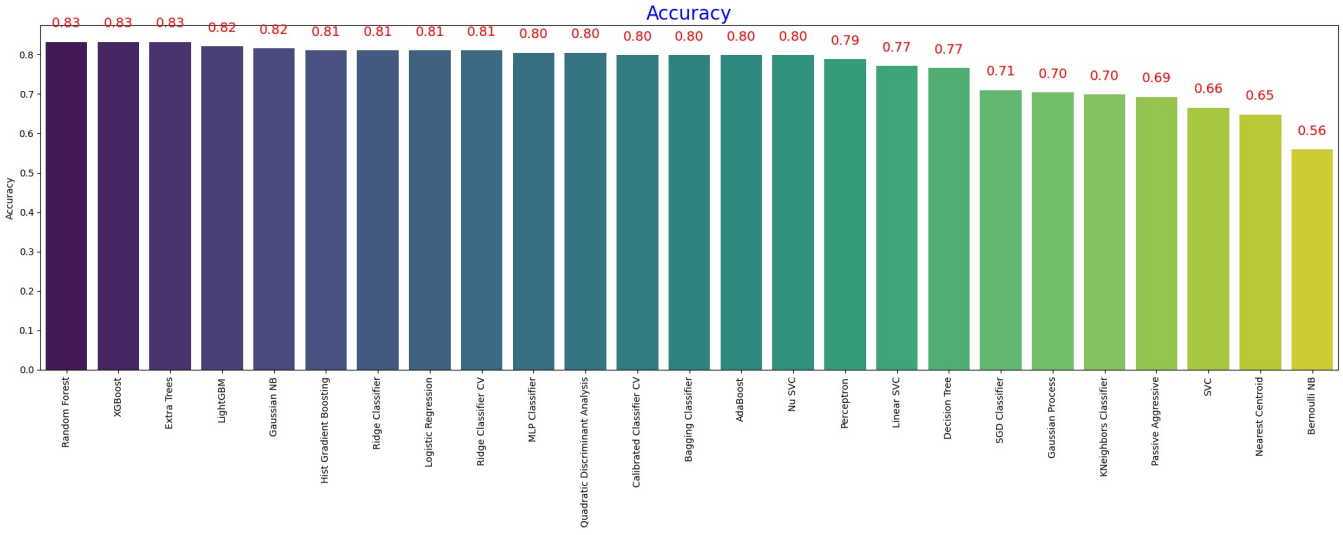
Additional Considerations:

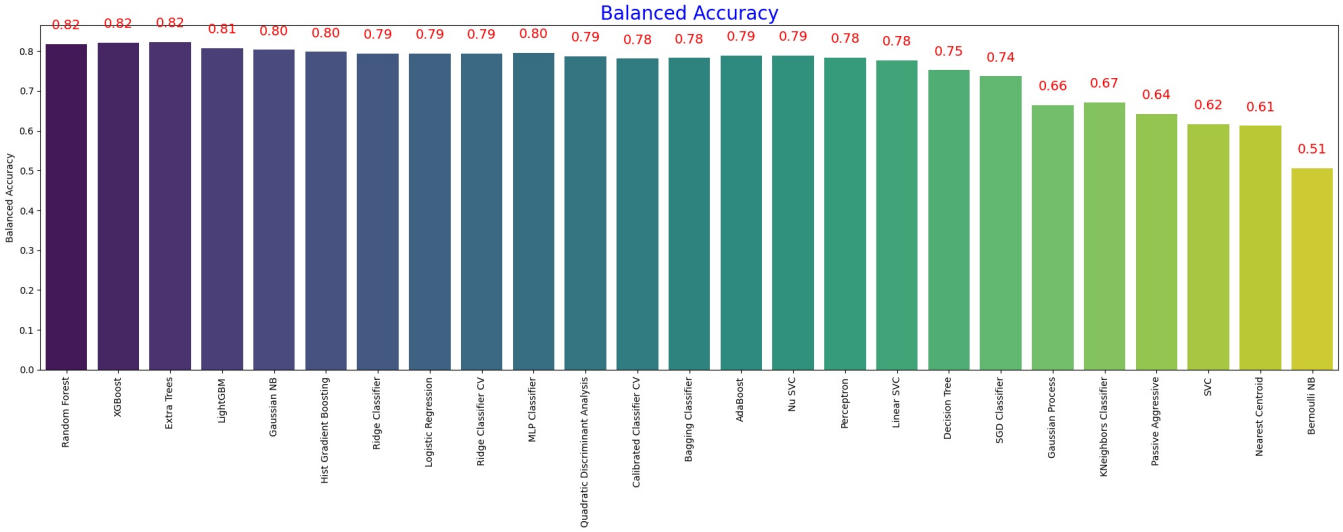
- **Multiple Metrics:** It's often beneficial to use multiple metrics to get a comprehensive understanding of the model's performance.
- **Domain Knowledge:** Leverage domain experts to understand the specific requirements and constraints of the problem.
- **Cross-Validation:** Use cross-validation to assess the model's generalization performance on unseen data.

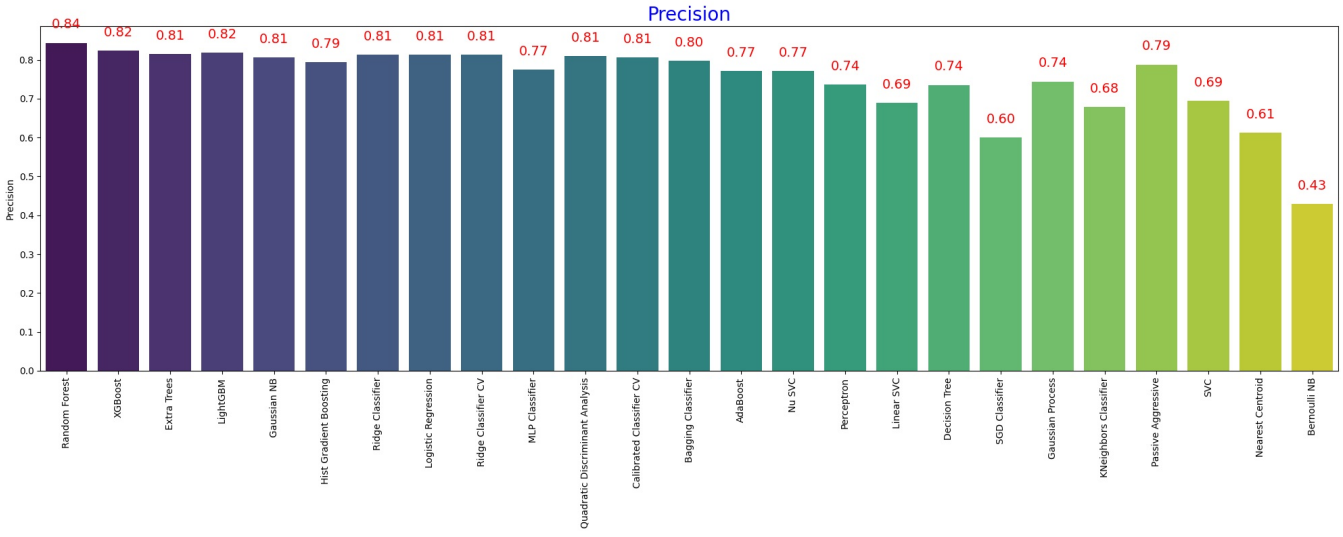
By carefully considering these factors and choosing appropriate metrics, you can obtain more generalized and reliable results from your classification models.

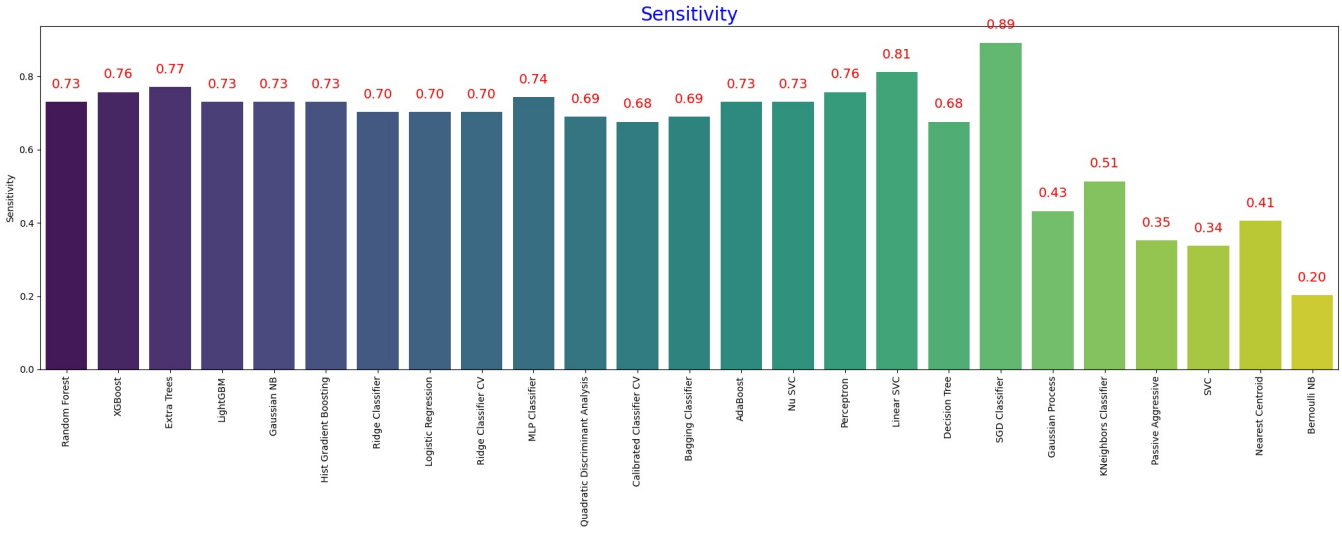
In [8]:
```
# List of metrics to plot
metrics = [
    'Accuracy',  # Assuming this is the column for model accuracy
    'Balanced Accuracy',
    'Precision',
    'Sensitivity',
    'Specificity',
    'F1 Score',
    'ROC AUC',
    'Log Loss',
    'Matthews Corrcoef'
]

# Loop through each metric and create a separate plot
for metric in metrics:
  # Create a new figure for each plot
  fig, ax = plt.subplots(figsize=(20, 8))  # Adjust figure size as needed

  # Create bar plot with Seaborn
  sns.barplot(x="Model", y=metric, data=model_results, palette="viridis", ax=ax)

  # Extract bar values (assuming a numerical column for metric values)
  bar_values = model_results[metric]

  # Add value annotations above each bar
  for i, (model, value) in enumerate(zip(model_results["Model"], bar_values)):
    ax.text(i, value + 0.03, f"{value:.2f}", ha='center', va='bottom', fontsize=14, color='red')

  # Set title and labels with increased font size and space before the plot
  ax.set_title(f"{metric}", fontsize=20, color='Blue')
  ax.set_xlabel("")
  ax.set_ylabel(f"{metric}")
```

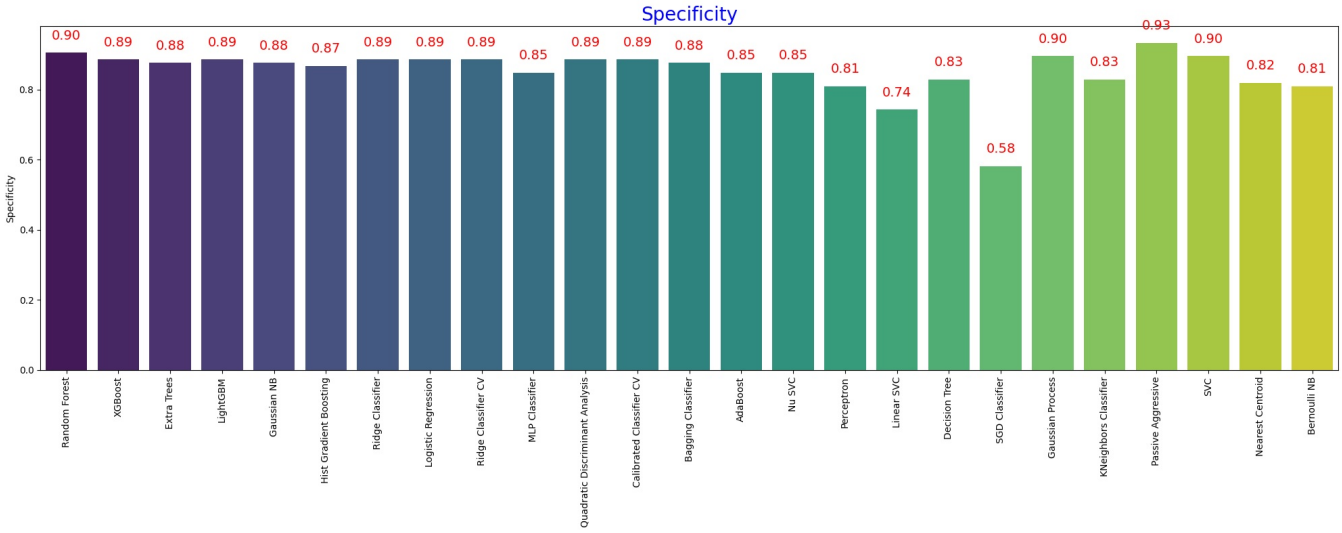
```
  # Increase space above the plot using `top` prop in `plt.subplots_adjust`
  plt.subplots_adjust(top=0.95)  # Adjust value as needed

  # Rotate x-axis labels if many models
  if len(model_results["Model"]) > 5:
    ax.set_xticklabels(model_results["Model"], rotation=90)

  # Adjust spacing within the plot
  plt.tight_layout()

# Show all plots at once
plt.show()
```

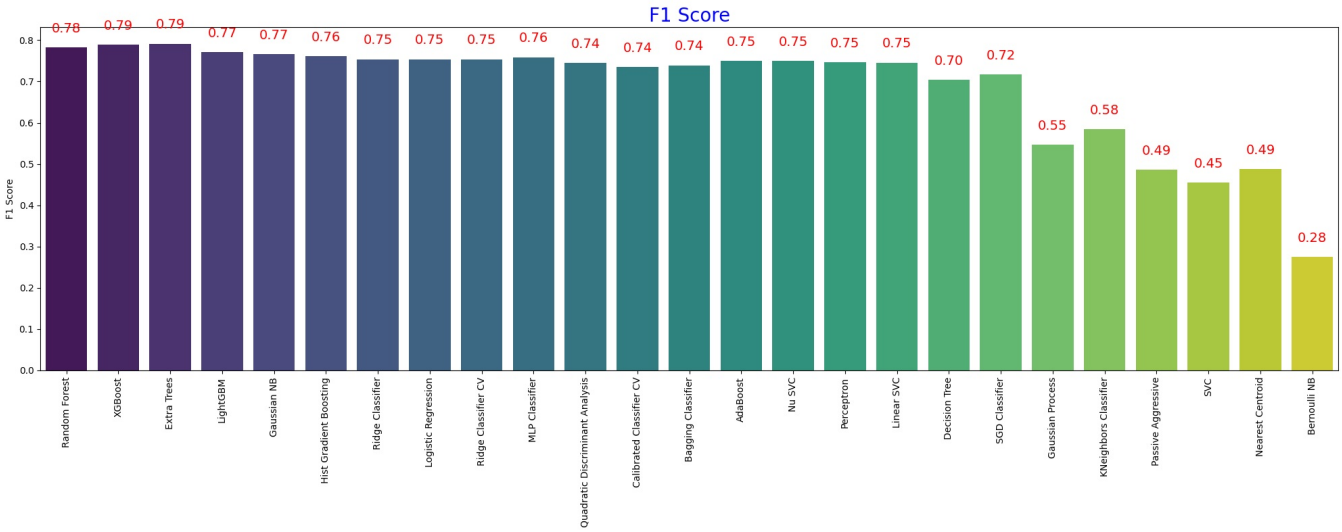


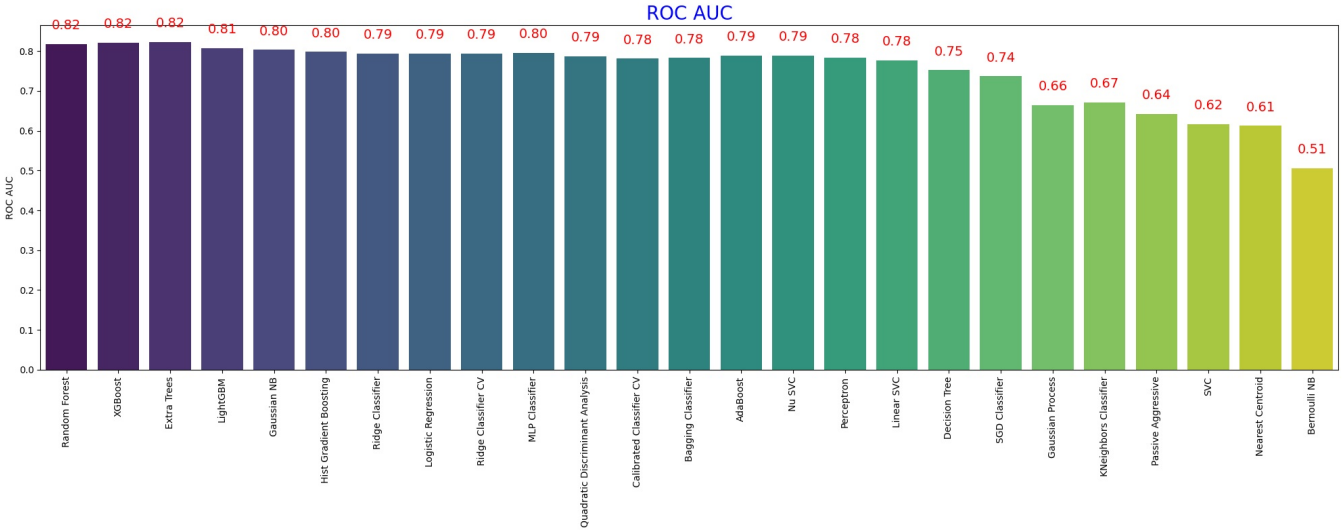


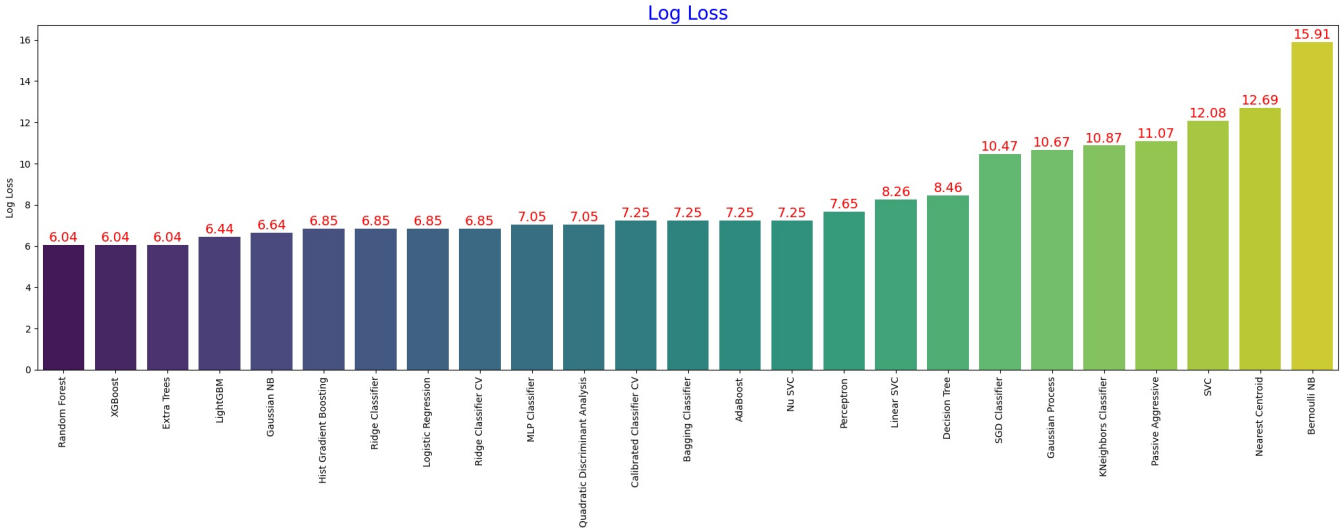


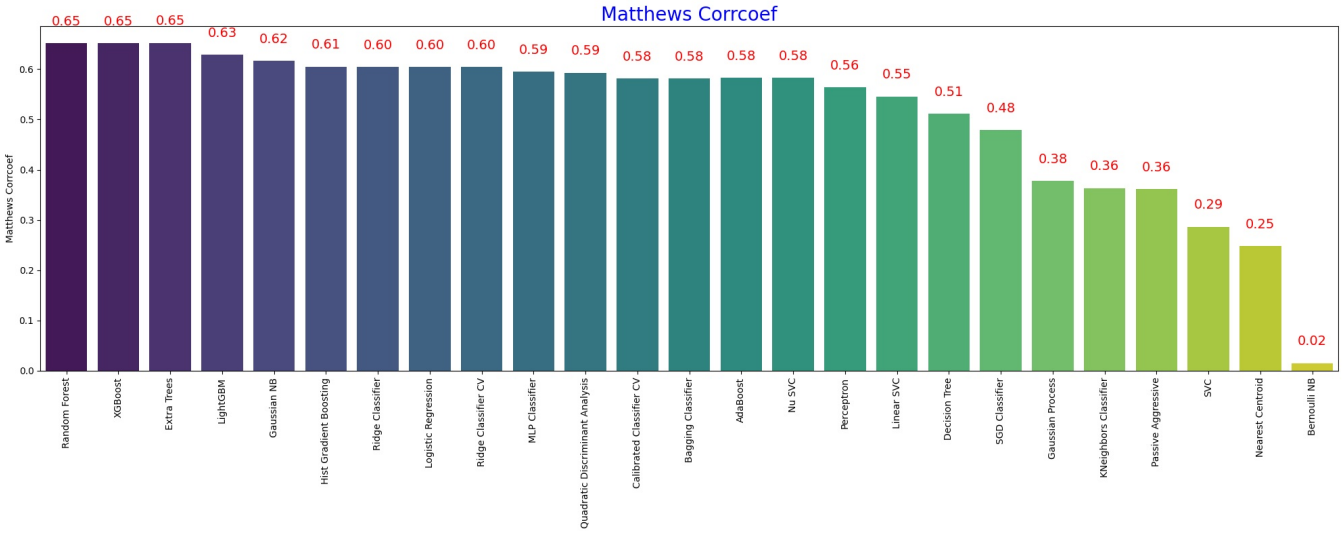

Sensitivity
Sensitivity
0.0
0.2
0.4
0.6
0.8
0.73
0.76
0.77
0.73
0.73
0.73
0.70
0.70
0.70
0.74
0.69
0.68
0.69
0.73
0.73
0.76
0.81
0.68
0.89
0.43
0.51
0.35
0.34
0.41
0.20
Random Forest
XGBoost
Extra Trees
LightGBM
Gaussian NB
Hist Gradient Boosting
Ridge Classifier
Logistic Regression
Ridge Classifier CV
MLP Classifier
Quadratic Discriminant Analysis
Calibrated Classifier CV
Bagging Classifier
AdaBoost
Nu SVC
Perceptron
Linear SVC
Decision Tree
SGD Classifier
Gaussian Process
KNeighbors Classifier
Passive Aggressive
SVC
Nearest Centroid
Bernoulli NB


Specificity
Specificity
0.0
0.2
0.4
0.6
0.8
0.90
0.89
0.88
0.89
0.88
0.87
0.89
0.89
0.89
0.85
0.89
0.89
0.88
0.85
0.85
0.81
0.74
0.83
0.58
0.90
0.83
0.93
0.90
0.82
0.81
Random Forest
XGBoost
Extra Trees
LightGBM
Gaussian NB
Hist Gradient Boosting
Ridge Classifier
Logistic Regression
Ridge Classifier CV
MLP Classifier
Quadratic Discriminant Analysis
Calibrated Classifier CV
Bagging Classifier
AdaBoost
Nu SVC
Perceptron
Linear SVC
Decision Tree
SGD Classifier
Gaussian Process
KNeighbors Classifier
Passive Aggressive
SVC
Nearest Centroid
Bernoulli NB


F1 Score
F1 Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.78
0.79
0.79
0.77
0.77
0.76
0.75
0.75
0.75
0.76
0.74
0.74
0.74
0.75
0.75
0.75
0.75
0.70
0.72
0.55
0.58
0.49
0.45
0.49
0.28
Random Forest
XGBoost
Extra Trees
LightGBM
Gaussian NB
Hist Gradient Boosting
Ridge Classifier
Logistic Regression
Ridge Classifier CV
MLP Classifier
Quadratic Discriminant Analysis
Calibrated Classifier CV
Bagging Classifier
AdaBoost
Nu SVC
Perceptron
Linear SVC
Decision Tree
SGD Classifier
Gaussian Process
KNeighbors Classifier
Passive Aggressive
SVC
Nearest Centroid
Bernoulli NB







https://www.kaggle.com/code/pythonafroz/votingclassifier-a-powerful-ensemble-technique/notebook

https://sites.google.com/view/aiml-deepthought/home

List of 30 common metrics used in evaluating Large Language Models (LLMs)

1. BLEU: Bilingual Evaluation Understudy
2. ROUGE: Recall-Oriented Understudy for Gisting Evaluation
3. METEOR: Metric for Evaluation of Translation with Explicit ORdering
4. TER: Translation Edit Rate
5. GLEU: Google-BLEU
6. CIDEr: Consensus-based Image Description Evaluation
7. SPICE: Semantic Propositional Image Caption Evaluation
8. BERTScore: BERT-based Scoring
9. BLEURT: BERT-based Language Understanding Evaluation Reference Tool
10. MAUVE: Measuring the Gap between Neural Text and Human Text
11. PPL: Perplexity
12. WER: Word Error Rate
13. CER: Character Error Rate
14. F1 Score: F1 Score (harmonic mean of precision and recall)
15. MRR: Mean Reciprocal Rank
16. NDCG: Normalized Discounted Cumulative Gain
17. MAP: Mean Average Precision
18. AUC-ROC: Area Under the Curve - Receiver Operating Characteristic
19. AUROC: Area Under the Receiver Operating Characteristic curve
20. MSE: Mean Squared Error
21. MAE: Mean Absolute Error
22. RMSE: Root Mean Square Error
23. MAPE: Mean Absolute Percentage Error
24. Accuracy: Accuracy

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

25. Precision: Precision
26. Recall: Recall
27. HTER: Human-targeted Translation Edit Rate
28. COMET: Crosslingual Optimized Metric for Evaluation of Translation
29. chrF: Character n-gram F-score
30. SacreBLEU: Standardized BLEU

This list covers many of the metrics used in LLM evaluation, but it's not exhaustive. The field is evolving rapidly, and new metrics are being developed to address specific aspects of language model performance.

1. BLEU: Bilingual Evaluation Understudy

BLEU (Bilingual Evaluation Understudy) is a metric used to evaluate the quality of machine-translated text. It compares a machine-generated translation to one or more human-created reference translations. The closer the machine translation is to the human-created references, the higher the BLEU score.

How BLEU Works

BLEU is based on the following intuition:

- **Precision:** How many of the words in the machine translation are also present in the reference translations?
- **Length Penalty:** Longer translations tend to have lower precision, so a penalty is applied to discourage overly long translations.

BLEU calculates a score based on these factors. A perfect match between the machine translation and the reference translations results in a BLEU score of 1.0, while a complete mismatch results in a score of 0.0.

Example

Let's say we have a sentence in English:

- **Reference:** "The quick brown fox jumps over the lazy dog."

And a machine-translated version:

- **Candidate:** "Quick brown fox jumps lazy dog."

To calculate the BLEU score, we would:

1. **Calculate n-gram precision:** Count the number of n-grams (sequences of n words) in the candidate that match n-grams in the reference. For example, the unigram precision would be 5/6 (5 matching words out of 6).
2. **Apply a length penalty:** If the candidate is too short or too long compared to the reference, a penalty is applied to discourage deviations from the reference length.
3. **Combine precision and length penalty:** The final BLEU score is calculated based on the weighted geometric mean of the n-gram precisions and the length penalty.

Limitations of BLEU

While BLEU is a widely used metric, it has some limitations:

- **It doesn't capture semantic meaning:** BLEU focuses on word-level matches, not the overall meaning of the translation.
- **It can be manipulated:** It's possible to create translations that score high on BLEU but are semantically incorrect.
- **It doesn't account for fluency:** BLEU doesn't evaluate how fluent or natural the machine translation sounds.

Despite these limitations, BLEU remains a valuable tool for evaluating machine translation systems, especially when used in conjunction with other metrics.

2. ROUGE: Recall-Oriented Understudy for Gisting Evaluation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate the quality of automatically generated summaries or translations. It compares a system-generated output (summary or translation) to one or more human-created reference summaries or translations. Unlike BLEU, which focuses on precision, ROUGE emphasizes recall, meaning it measures how well the system output covers the information present in the reference summaries.

How ROUGE Works

ROUGE calculates various metrics based on n-gram overlap between the system output and reference summaries. N-grams are sequences of n words. Here are some common ROUGE metrics:

- **ROUGE-N:** Calculates the n-gram overlap between the system and reference summaries. For example, ROUGE-1 considers unigrams (single words), ROUGE-2 considers bigrams (pairs of words), and so on.
- **ROUGE-L:** Measures the longest common subsequence (LCS) between the system and reference summaries.
- **ROUGE-W:** Similar to ROUGE-L but incorporates weights for different n-gram lengths.

Example

Let's consider a text summarization task.

- **Reference summary:** "The quick brown fox jumps over the lazy dog."
- **System summary:** "Quick brown fox jumps."

- **ROUGE-1:** Overlap of unigrams: 4/5 = 0.8
- **ROUGE-2:** Overlap of bigrams: 2/3 = 0.67

A higher ROUGE score indicates better overlap between the system output and reference summaries.

Advantages of ROUGE

- Simple to compute
- Widely used in the evaluation of text summarization and machine translation systems

Limitations of ROUGE

- Focuses on n-gram matching, which might not capture semantic similarity
- Doesn't consider the ordering of words in the summary
- Can be easily manipulated by generating redundant summaries

Conclusion

ROUGE is a valuable tool for evaluating text summarization and machine translation systems, but it should be used in conjunction with other metrics to get a comprehensive assessment of the system's quality.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

3. METEOR: Metric for Evaluation of Translation with Explicit Ordering

METEOR (Metric for Evaluation of Translation with Explicit Ordering) is a metric used to evaluate the quality of machine-translated text. It is designed to address some of the limitations of other metrics like BLEU. Unlike BLEU, which primarily focuses on n-gram precision, METEOR incorporates both precision and recall, along with other factors.

METEOR operates in the following steps:

1. **Tokenization:** Both the machine-translated output and the reference translation are tokenized into words or subwords.
2. **Matching:** Words from the machine translation are matched to the reference translation based on exact matches, stemming, and synonymy.
3. **Alignment:** The matched words are aligned to create a mapping between the two sentences.
4. **Penalty:** A penalty is applied based on the fragmentation of the matched words. More fragmented matches result in a higher penalty.
5. **Calculation:** METEOR calculates a score based on the harmonic mean of unigram precision and recall, and the penalty.

Key Features of METEOR

- **Combines precision and recall:** This provides a more balanced evaluation of the translation.
- **Considers word order:** By aligning matched words, METEOR takes into account the order of words in the sentence.
- **Incorporates stemming and synonymy:** This allows for more flexible matching, improving the metric's robustness.

- **Penalty for fragmentation:** Penalizes translations with highly fragmented matches, promoting more fluent translations.

Example

- **Reference:** "The quick brown fox jumps over the lazy dog."
- **Machine Translation:** "A fast brown fox jumps over the lazy dog."

METEOR would:

- Match "quick" with "fast" using synonymy.
- Match "brown", "fox", "jumps", "over", "the", "lazy", and "dog" as exact matches.
- Align the matched words to create an alignment.
- Calculate a penalty based on the alignment, considering the substitution of "quick" with "fast".
- Compute the final METEOR score based on precision, recall, and the penalty.

Advantages of METEOR

- Better correlation with human judgment than BLEU
- Considers both precision and recall
- Accounts for word order and semantic similarity

Limitations of METEOR

- More computationally expensive than BLEU
- Still relies on n-gram matching and might not capture all aspects of translation quality

In summary, METEOR offers a more comprehensive evaluation of machine translation quality compared to BLEU. By considering word order, synonymy, and fragmentation, it provides a better approximation of human judgment.

4. TER: Translation Edit Rate

TER (Translation Edit Rate) is a metric used to evaluate the quality of machine translation. Unlike metrics like BLEU which focus on n-gram overlap, TER measures the amount of editing required to transform a machine-translated output into a human-quality reference translation.

How TER Works

1. **Tokenization:** Both the machine translation and reference translations are broken down into words or subwords.
2. **Alignment:** An alignment is created between the machine translation and the reference translation. This alignment identifies corresponding words or phrases between the two.
3. **Edit Operations:** The minimum number of edit operations (insertions, deletions, substitutions, and shifts) required to transform the machine translation into the reference translation is calculated.
4. **Normalization:** The total number of edit operations is divided by the average length of the reference translations to obtain the TER score.

A lower TER score indicates a better translation, as it implies fewer edits are needed to make it correct.

Example

Consider the following:

- **Reference:** "The quick brown fox jumps over the lazy dog."
- **Machine Translation:** "Quick brown fox jumps lazy dog."

To calculate TER, we would:

1. Identify the missing word "the" (insertion).
2. Calculate the total edit operations: 1 (insertion).
3. Calculate the average length of the reference (assuming only one reference): 7 words.
4. TER = 1 / 7 = 0.143

Advantages of TER

- Correlates well with human judgment of translation quality.
- Directly measures the effort required to post-edit a machine translation.
- Can be used to identify specific error types in machine translation.

Limitations of TER

- More computationally expensive than some other metrics.
- Sensitive to the choice of alignment algorithm.

In summary, TER provides a more intuitive measure of translation quality by directly assessing the editing effort required to correct the machine output. It is a valuable metric for evaluating machine translation systems and guiding improvement efforts.

5. GLEU: Google BLEU

GLEU (Google BLEU) is a variation of the BLEU metric specifically designed for evaluating single sentences rather than entire corpora. It addresses some limitations of the original BLEU metric when applied to individual sentences.

How GLEU Works

1. **N-gram extraction:** All possible sub-sequences of 1, 2, 3, or 4 tokens (n-grams) are extracted from both the reference and the generated sentence.
2. **Precision and Recall calculation:**
    - **Precision:** The ratio of matching n-grams in the generated sentence to the total number of n-grams in the generated sentence.
    - **Recall:** The ratio of matching n-grams in the generated sentence to the total number of n-grams in the reference sentence.
3. **GLEU score:** The GLEU score is calculated as the minimum of precision and recall.

Example

Consider the following:

- **Reference:** "The quick brown fox jumps over the lazy dog."
- **Generated:** "Quick brown fox jumps lazy dog."

To calculate GLEU:

1. Extract n-grams from both sentences.
2. Calculate precision and recall for each n-gram.
3. The GLEU score is the minimum of the calculated precision and recall values across all n-grams.

Key Differences from BLEU

- **Focus on single sentences:** GLEU is specifically designed for evaluating individual sentences, while BLEU is primarily used for evaluating entire corpora.
- **Calculation of precision and recall:** GLEU calculates precision and recall based on n-gram matches, while BLEU uses a more complex formula involving geometric mean and length penalty.

Advantages of GLEU

- Better suited for evaluating single sentence translations.
- Simpler calculation compared to BLEU.
- Provides a more intuitive measure of similarity between the generated and reference sentences.

Limitations of GLEU

- Might not capture all nuances of translation quality.
- Less widely used than BLEU.

In summary, GLEU offers a simpler and more direct approach to evaluating single sentence translations compared to BLEU. While it might not be as comprehensive as BLEU, it can be a useful metric in certain scenarios.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

6. CIDEr: Consensus-Based Image Description Evaluation

(Consensus-Based Image Description Evaluation) is a metric designed to evaluate the quality of image descriptions generated by computer models. Unlike other metrics that primarily focus on n-gram overlap or lexical similarity, CIDEr emphasizes the agreement between a generated description and multiple human-written reference descriptions.

How CIDEr Works

1. **N-gram extraction:** Similar to BLEU, CIDEr extracts n-grams (sequences of words) from both the generated description and the reference descriptions.
2. **TF-IDF weighting:** The extracted n-grams are weighted using TF-IDF (Term Frequency-Inverse Document Frequency) to account for the importance of different words in the reference descriptions.
3. **Consensus calculation:** CIDEr computes the cosine similarity between the TF-IDF vectors of the generated description and each reference description. The average of these similarities is taken as the final CIDEr score.

Key Features of CIDEr

- **Focus on consensus:** CIDEr directly measures how well a generated description aligns with multiple human-written descriptions.
- **TF-IDF weighting:** This helps to capture the semantic relevance of words in the evaluation.
- **Correlation with human judgment:** CIDEr has shown a strong correlation with human ratings of image description quality.

Example

Imagine an image of a cat sitting on a mat.

- **Reference descriptions:**
  - "A tabby cat is sitting on a blue mat."
  - "A cat is sleeping on a mat."
  - "A furry cat is resting on a blue mat."
- **Generated description:** "A cat is on a mat."

CIDEr would calculate the similarity between the generated description and each reference description based on n-gram overlap and TF-IDF weights. The final CIDEr score would reflect how well the generated description aligns with the consensus among the human-written descriptions.

Advantages of CIDEr

- Better captures human judgment of image description quality compared to other metrics.
- Considers semantic similarity through TF-IDF weighting.

Limitations of CIDEr

- More computationally expensive than some other metrics.
- Relies on the quality of the reference descriptions.

In summary, CIDEr is a valuable metric for evaluating image description models as it aligns well with human perception of quality. By considering consensus and semantic similarity, it provides a more comprehensive evaluation than metrics that focus solely on lexical overlap.

7. SPICE: Semantic Propositional Image Caption Evaluation

SPICE (Semantic Propositional Image Caption Evaluation) is a metric designed to evaluate the semantic quality of image captions generated by computer models. Unlike metrics like BLEU and CIDEr, which focus on n-gram overlap or lexical similarity, SPICE focuses on the underlying semantic meaning of the caption.

How SPICE Works

1. **Scene graph generation:** Both the generated caption and the reference captions are converted into scene graphs. A scene graph represents an image as a set of objects, their attributes, and relationships between them.
2. **Semantic proposition matching:** The scene graphs of the generated and reference captions are compared based on their semantic propositions (e.g., "person holding ball").
3. **Matching score calculation:** A matching score is calculated based on the number of matched semantic propositions and their attributes.
4. **SPICE score:** The final SPICE score is computed as the average matching score across all reference captions.

Key Features of SPICE

- **Focus on semantic meaning:** SPICE directly evaluates the underlying semantic content of the caption.
- **Scene graph representation:** By using scene graphs, SPICE captures complex relationships between objects.
- **Correlation with human judgment:** SPICE has shown a strong correlation with human ratings of image caption quality.

Example

Imagine an image of a person holding a red ball.

- **Reference captions:**
  - "A person is holding a red ball."
  - "The man is playing with a red ball."
- **Generated caption:** "A woman is holding a ball."

SPICE would convert each caption into a scene graph, identifying objects (person, ball), attributes (red), and relationships (holding). The scene graphs of the generated and reference captions would then be compared to calculate the matching score based on shared semantic propositions.

Advantages of SPICE

- Better captures the semantic meaning of image captions compared to other metrics.
- Handles variations in language and syntax.
- Correlates well with human judgment.

Limitations of SPICE

- More complex and computationally expensive than other metrics.
- Requires accurate scene graph generation.

In summary, SPICE provides a more robust and informative evaluation of image captions by focusing on semantic content rather than superficial lexical matching. It is a valuable tool for assessing the quality of image captioning models.

8. BERTScore: BERT-Based Scoring

BERTScore is a metric used to evaluate the quality of text generation tasks, such as machine translation, text summarization, and question answering. Unlike traditional metrics like BLEU and ROUGE, which rely on n-gram matching, BERTScore leverages pre-trained language models, specifically BERT, to capture semantic and syntactic similarities between text pairs.

How BERTScore Works

1. **Token Embedding:** Both the reference and candidate text are tokenized, and each token is converted into a contextual embedding using a pre-trained BERT model.
2. **Matching:** Each token in the candidate text is matched with the most similar token in the reference text based on cosine similarity between their embeddings.
3. **Precision, Recall, and F1:**
    - **Precision:** Measures how many tokens in the candidate text are correctly matched to the reference text.
    - **Recall:** Measures how many tokens in the reference text are correctly matched by the candidate text.
    - **F1:** The harmonic mean of precision and recall.
4. **Scoring:** The final BERTScore is computed as the average of the F1 scores across all tokens in the candidate text.

Example

Consider the following:

- **Reference:** "The quick brown fox jumps over the lazy dog."
- **Candidate:** "A fast brown fox jumps over the sleepy dog."

BERTScore would:

- Convert each word into a BERT embedding.
- Match similar words based on cosine similarity (e.g., "quick" and "fast").
- Calculate precision, recall, and F1 for each matched token.
- Compute the average F1 score as the final BERTScore.

Advantages of BERTScore

- Captures semantic and syntactic similarities between text pairs.
- Correlates better with human judgment than traditional metrics.
- Works well for various NLP tasks.

Limitations of BERTScore

- Computationally more expensive than traditional metrics.
- Relies on the quality of the pre-trained language model.

In summary, BERTScore offers a more sophisticated and accurate way to evaluate text generation tasks by leveraging the power of pre-trained language models. It is a valuable tool for assessing the quality of generated text and guiding model improvement.

9. BLEURT: BERT-based Language Understanding Evaluation Reference Tool

BLEURT is a state-of-the-art metric for evaluating the quality of generated text. It builds upon the success of BERT, a powerful language model, to provide a more robust and human-like assessment of text generation systems. Unlike traditional metrics like BLEU, which primarily focus on n-gram overlap, BLEURT leverages the deep linguistic understanding of BERT to capture nuances in language and meaning.

How BLEURT Works

1. **Pre-training:** BLEURT is pre-trained on a massive dataset of human-quality text, allowing it to learn a rich representation of language.
2. **Evaluation:** Given a generated text and a reference text, BLEURT computes a similarity score between the two. This score represents the model's assessment of how closely the generated text matches the quality of human-written text.

Key Features of BLEURT

- **Leverages BERT:** BLEURT benefits from BERT's ability to understand complex language patterns and semantic relationships.
- **Pre-trained on large datasets:** This enables BLEURT to capture a wide range of linguistic phenomena.
- **Strong correlation with human judgment:** BLEURT has demonstrated a high correlation with human ratings of text quality.

Example

Consider the following:

- **Reference:** "The quick brown fox jumps over the lazy dog."
- **Generated:** "A fast brown fox leaps over the sleepy dog."

BLEURT would process both sentences using its pre-trained BERT model, capturing the semantic and syntactic similarities between the two. It would then output a score indicating how close the generated text is to the quality of the reference text.

Advantages of BLEURT

- Better correlates with human judgment than traditional metrics.
- Captures nuances of language and meaning.
- Robust to different text generation tasks.

Limitations of BLEURT

- Requires significant computational resources for training and evaluation.
- Might be sensitive to the quality of the pre-training data.

In summary, BLEURT is a powerful tool for evaluating the quality of generated text. By leveraging the capabilities of BERT, it provides a more accurate and human-like assessment than traditional metrics.

10. MAUVE: Measuring the Gap Between Neural Text and Human Text

MAUVE (Measuring the Gap Between Neural Text and Human Text) is a metric designed to assess the quality of text generated by neural models. It measures how closely the generated text distribution matches the distribution of human-written text. Unlike traditional metrics that focus on specific aspects like n-gram overlap or lexical similarity, MAUVE provides a more holistic evaluation.

How MAUVE Works

1. **Embedding:** Both the human-written text and the generated text are converted into embeddings using a pre-trained language model.
2. **Quantization:** The embeddings are quantized into a fixed number of clusters using k-means clustering. This reduces the dimensionality of the data and makes the calculations more efficient.
3. **Divergence Calculation:** The Kullback-Leibler (KL) divergence is calculated between the probability distributions of the quantized embeddings for the human and generated text. This measures how different the two distributions are.
4. **Divergence Curve:** By varying the number of clusters (quantization level), MAUVE creates a divergence curve that captures the trade-off between different types of errors in the generated text.
5. **MAUVE Score:** The final MAUVE score is the area under the divergence curve, providing a single scalar value to represent the overall similarity between the two text distributions.

Key Features of MAUVE

- **Focus on distribution:** MAUVE directly compares the overall distribution of generated text to human text, capturing a broader range of quality aspects.
- **Quantization:** This technique makes the computation efficient and allows for analysis at different levels of granularity.
- **Divergence curve:** Provides insights into the specific types of errors in the generated text.

Example

Imagine comparing the text generated by a language model to a corpus of human-written articles. MAUVE would:

1. Convert both the generated and human text into embeddings.
2. Quantize these embeddings into a set of clusters.
3. Calculate the KL divergence between the distributions of the quantized embeddings for the two text sets.
4. Create a divergence curve by varying the number of clusters.
5. Compute the area under the divergence curve to obtain the final MAUVE score.

A higher MAUVE score indicates that the generated text distribution is closer to the human text distribution, suggesting better quality.

Advantages of MAUVE

- Provides a more comprehensive evaluation of text generation quality than traditional metrics.

- Sensitive to different types of errors in generated text.
- Correlates well with human judgment.

Limitations of MAUVE

- Computationally more expensive than some other metrics.
- Requires careful selection of hyperparameters for quantization.

In summary, MAUVE offers a valuable tool for assessing the quality of text generated by neural models. By focusing on the overall distribution of text, it provides a more nuanced evaluation than traditional metrics.

11. Perplexity (PPL)

Perplexity is a metric used to evaluate the performance of language models. It measures how well a probability model predicts a sample. In simpler terms, it quantifies how surprised a language model is when it encounters new text.

- **Probability distribution:** A language model assigns probabilities to sequences of words. For example, given the sentence "The quick brown fox", the model assigns a probability to the word "jumps" appearing next.
- **Negative log-likelihood:** The negative log-likelihood of a sequence is calculated by multiplying the probabilities of each word in the sequence and then taking the negative logarithm of the result.
- **Average negative log-likelihood:** This is the average of the negative log-likelihoods of all sequences in a dataset.
- **Perplexity:** The perplexity is the exponentiation of the average negative log-likelihood.

Formula:

$$PPL(X) = \exp(-(1/T) * \Sigma \log p(xi|x1, ..., xi-1))$$

- PPL(X) is the perplexity of the sequence X
- T is the length of the sequence
- p(xi|x1, ..., xi-1) is the probability of the ith word given the previous words

Interpretation

- **Lower perplexity is better:** A lower perplexity indicates that the model is better at predicting the next word in a sequence, suggesting a better understanding of the language.

- **Perplexity as a branching factor:** Perplexity can be interpreted as the average number of choices a language model considers at each position in a text. A lower perplexity means the model has fewer choices, indicating better confidence.

Example

Imagine two language models, A and B. When presented with the sentence "The quick brown fox", model A assigns higher probabilities to common word sequences like "jumps over", while model B assigns higher probabilities to less common sequences. Model A will likely have a lower perplexity, indicating it is better at capturing language patterns.

In essence, perplexity provides a quantitative measure of how well a language model captures the underlying patterns and structure of human language.

12. Word Error Rate (WER)

Word Error Rate (WER) is a metric used to evaluate the accuracy of speech recognition or machine translation systems. It measures the number of errors (insertions, deletions, and substitutions) in a generated text compared to a reference text. A lower WER indicates higher accuracy.

How WER is Calculated

To calculate WER, you need to determine the number of insertions, deletions, and substitutions.

- **Insertion:** A word is added in the generated text that doesn't exist in the reference text.
- **Deletion:** A word is missing in the generated text that exists in the reference text.
- **Substitution:** A word is replaced by a different word in the generated text.

The formula for WER is:

WER = (S + D + I) / N

Where:

- S = number of substitutions
- D = number of deletions
- I = number of insertions
- N = total number of words in the reference text

Example

Reference text: "The quick brown fox jumps over the lazy dog" **Generated text:** "The quick brown fox jumps over the dog"

In this example:

- There is one deletion: "lazy"
- There are no insertions or substitutions

So, WER = (0 + 1 + 0) / 9 = 1/9 = 0.11 or 11%

Interpretation: The speech recognition system made an 11% error in transcribing the sentence.

Note:

- WER is often used in conjunction with other metrics like Character Error Rate (CER) for a more comprehensive evaluation.
- While WER is a common metric, it has limitations. For example, it doesn't consider semantic similarity between words.

13. Character Error Rate (CER)

Character Error Rate (CER) is a metric used to evaluate the accuracy of systems that process text, such as Optical Character Recognition (OCR) or Speech-to-Text systems. It measures the number of character errors (insertions, deletions, and substitutions) in a generated text compared to a reference text. A lower CER indicates higher accuracy.

How CER is calculated

To calculate CER, you need to determine the number of insertions, deletions, and substitutions.

- **Insertion:** A character is added in the generated text that doesn't exist in the reference text.
- **Deletion:** A character is missing in the generated text that exists in the reference text.
- **Substitution:** A character is replaced by a different character in the generated text.

The formula for CER is:

CER = (S + D + I) / N

Where:

- S = number of substitutions
- D = number of deletions
- I = number of insertions
- N = total number of characters in the reference text

Example

Reference text: "The quick brown fox" **Generated text:** "Th equik brwn foxx"

In this example:

- There is one deletion: "e"
- There are two substitutions: "u" for "i" and "x" for "o"
- There is one insertion: "x"

So, CER = (2 + 1 + 1) / 17 = 4 / 17 ≈ 0.235 or 23.5%

Interpretation: The system made a 23.5% character error in transcribing the text.

Note:

- CER is often used in conjunction with Word Error Rate (WER) for a more comprehensive evaluation.
- While CER is a useful metric, it doesn't consider the semantic or syntactic structure of the text.

14. F1 Score: A Balance of Precision and Recall

The F1 score is a metric used to evaluate the performance of a model, especially in classification tasks. It combines precision and recall into a single value.

Precision and Recall

Before we dive into F1 score, let's quickly recap precision and recall:

- **Precision:** Measures the accuracy of positive predictions. It is the ratio of true positives to the sum of true positives and false positives.
- **Recall:** Measures the ability of a model to find all positive cases. It is the ratio of true positives to the sum of true positives and false negatives.

F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a single metric to evaluate a model's performance.

Formula:

F1 Score = 2 * (precision * recall) / (precision + recall)

An F1 score ranges from 0 to 1, with 1 being the best possible value.

Example

Imagine a model that predicts whether an email is spam or not.

- **True Positives (TP):** Emails correctly classified as spam.
- **True Negatives (TN):** Emails correctly classified as not spam.
- **False Positives (FP):** Non-spam emails incorrectly classified as spam.
- **False Negatives (FN):** Spam emails incorrectly classified as not spam.

Let's say:

- TP = 80
- FP = 20
- FN = 10

Precision = TP / (TP + FP) = 80 / (80 + 20) = 0.8 Recall = TP / (TP + FN) = 80 / (80 + 10) = 0.889

F1 Score = 2 * (0.8 * 0.889) / (0.8 + 0.889) ≈ 0.843

Why Harmonic Mean?

The harmonic mean is used instead of the arithmetic mean because it gives more weight to lower values. If either precision or recall is low, the F1 score will also be low. This ensures that the model performs well in both aspects.

In summary, the F1 score provides a balanced evaluation of a model's performance by considering both precision and recall. It's a valuable metric for assessing the overall effectiveness of a classification model

15. Mean Reciprocal Rank (MRR)

Mean Reciprocal Rank (MRR) is a metric used to evaluate the performance of ranking systems, such as search engines or recommender systems. It measures the average position of the first correct answer in a ranked list of results.

How MRR works

1. **Reciprocal Rank (RR):** For each query, the reciprocal rank is calculated as 1 divided by the position of the first correct answer.
    - If the correct answer is in the first position, RR = 1/1 = 1.
    - If the correct answer is in the second position, RR = 1/2 = 0.5.
    - If no correct answer is found, RR = 0.
2. **Mean Reciprocal Rank (MRR):** The MRR is the average of the reciprocal ranks across all queries.

Example

Consider a search engine that returns a ranked list of results for a query.

- **Query 1:** Correct answer at position 3. RR = 1/3.
- **Query 2:** Correct answer at position 1. RR = 1/1 = 1.
- **Query 3:** No correct answer found. RR = 0.

MRR = (1/3 + 1 + 0) / 3 = 4/9 ≈ 0.44

Interpretation

- MRR ranges from 0 to 1.
- A higher MRR indicates better performance, as it means the correct answer is found on average in a higher position.

Key points to remember:

- MRR only considers the position of the first correct answer, ignoring subsequent correct answers.
- MRR is sensitive to outliers, as a single very low rank can significantly impact the overall score.

Other metrics like Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) can provide more granular information about ranking performance.

16. Normalized Discounted Cumulative Gain (NDCG)

NDCG is a metric used to evaluate the quality of a ranked list of items, such as search results or recommendations. It considers both the relevance of each item and its position in the list.

How NDCG works

1. **Discounted Cumulative Gain (DCG):**
    - Assigns a relevance score to each item in the ranked list.
    - Discounts the relevance score of items lower in the list.
    - Calculates the sum of the discounted relevance scores.
2. **Ideal Discounted Cumulative Gain (IDCG):**
    - Calculates the DCG for a perfect ranking of the items (highest relevance items at the top).
3. **NDCG:**
    - Divides the DCG by the IDCG to normalize the score between 0 and 1.

Example

Imagine a search engine returning results for the query "best smartphones". We assign relevance scores to each result (4 for most relevant, 3 for less relevant, etc.).

Rank	Result	Relevance
1	Phone A	4
2	Phone B	3
3	Phone C	2

Export to Sheets

DCG:

- DCG = 4 + (3 / log2(2)) + (2 / log2(3)) ≈ 6.21

IDCG:

- Assuming Phone A is the most relevant, Phone B the second, and Phone C the third:
- IDCG = 4 + (3 / log2(2)) + (2 / log2(3)) ≈ 6.21 (same as DCG in this case, as the list is already perfectly ordered)

NDCG:

- NDCG = DCG / IDCG = 6.21 / 6.21 = 1

In this example, the NDCG is 1, indicating a perfect ranking.

Key points

- NDCG considers both relevance and position of items in the list.
- It is normalized, allowing comparison across different datasets and query lengths.
- Higher NDCG values indicate better ranking quality.

NDCG is widely used in information retrieval, recommendation systems, and other areas where ranking is important.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

17. Mean Average Precision (MAP)

Mean Average Precision (MAP) is a metric used to evaluate the performance of information retrieval systems, such as search engines or recommender systems. It measures the average precision of a system across multiple queries.

How MAP works

1. **Average Precision (AP):** For each query, calculate the precision at each relevant result retrieved. Then, calculate the average of these precision values.
2. **Mean Average Precision (MAP):** Calculate the average of the AP values for all queries.

Example

Consider a search engine with three queries: "apple", "banana", and "orange".

- **Query: apple**
  - Relevant results: A, B, C, D
  - Retrieved results: A, C, E, F
  - Precision at each relevant result: 1/1, 2/3, 3/4, 4/5
  - Average Precision (AP) for "apple" = (1 + 2/3 + 3/4 + 4/5) / 4
- **Query: banana**
  - Relevant results: X, Y, Z
  - Retrieved results: X, W, Y
  - Average Precision (AP) for "banana" = (1/1 + 2/3) / 3
- **Query: orange**
  - Relevant results: P, Q
  - Retrieved results: R, S
  - Average Precision (AP) for "orange" = 0

- **MAP:** Calculate the average of the AP values for "apple", "banana", and "orange".

Interpretation

- MAP ranges from 0 to 1.
- A higher MAP indicates better performance, as it means the system is better at retrieving relevant results in higher positions.

Key points to remember:

- MAP considers the order of relevant results in the ranked list.
- MAP is more informative than simple precision or recall as it considers the entire ranked list.

MAP is widely used in information retrieval and recommendation systems to evaluate the overall performance of a system.

18. AUC-ROC: Area Under the Curve - Receiver Operating Characteristic

Understanding ROC and AUC

ROC stands for Receiver Operating Characteristic. It's a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

- **True Positive Rate (TPR)** or sensitivity is the proportion of actual positives that are correctly identified as such.
- **False Positive Rate (FPR)** or fall-out is the proportion of actual negatives that are incorrectly identified as positives.

AUC stands for Area under the Curve. It is a numerical value representing the two-dimensional area under the ROC curve.

How to Calculate AUC-ROC

1. **Predict probabilities:** The model predicts the probability of each data point belonging to the positive class.
2. **Create various thresholds:** Different classification thresholds are set to classify instances as positive or negative.
3. **Calculate TPR and FPR:** For each threshold, calculate the TPR and FPR.
4. **Plot ROC curve:** Plot the TPR against the FPR for different threshold values.
5. **Calculate AUC:** Calculate the area under the ROC curve.

Interpretation of AUC-ROC

- **AUC ranges from 0 to 1.**
- **An AUC of 1** means the classifier perfectly distinguishes between positive and negative classes.
- **An AUC of 0.5** means the classifier is no better than random guessing.
- **Higher AUC values indicate better performance.**

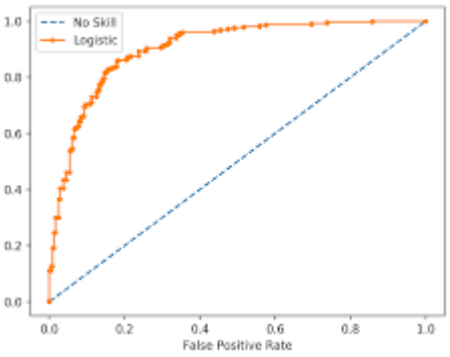
Example

Imagine a model predicting whether an email is spam or not.

- **True Positives:** Spam emails correctly identified as spam.
- **True Negatives:** Non-spam emails correctly identified as not spam.
- **False Positives:** Non-spam emails incorrectly identified as spam.
- **False Negatives:** Spam emails incorrectly identified as not spam.

By varying the classification threshold, you can calculate TPR and FPR for different scenarios. Plotting these points creates the ROC curve. The area under this curve is the AUC.

Visual Representation



ROC curve

Advantages of AUC-ROC

- **Independent of class distribution:** Unlike accuracy, AUC is not affected by imbalanced datasets.
- **Considers all classification thresholds:** Provides a comprehensive view of the model's performance.

Limitations of AUC-ROC

- **Doesn't directly optimize a specific metric:** AUC doesn't directly optimize precision, recall, or F1-score.

In conclusion, AUC-ROC is a valuable metric for evaluating the performance of binary classification models. It provides a comprehensive understanding of the model's ability to distinguish between positive and negative classes.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

19. AUROC: Area Under the Receiver Operating Characteristic Curve

AUROC is a performance metric for classification models. It stands for **Area Under the Receiver Operating Characteristic Curve**.

- **ROC Curve:** A graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.
- **AUC:** The area under the ROC curve. A higher AUC indicates better performance.

How it works

1. **Model Predictions:** A classification model predicts the probability of each data point belonging to the positive class.
2. **Thresholding:** Different classification thresholds are applied to convert probabilities into binary predictions (positive or negative).
3. **Calculating TPR and FPR:** For each threshold, calculate the True Positive Rate (TPR) and False Positive Rate (FPR).
    - **TPR:** Proportion of actual positives correctly identified as such.
    - **FPR:** Proportion of actual negatives incorrectly identified as positives.
4. **Plotting the ROC Curve:** Plot the TPR against the FPR for different threshold values.
5. **Calculating AUC:** Calculate the area under the ROC curve.

Example: Fraud Detection

Imagine a model predicting whether a transaction is fraudulent or not.

- **True Positive:** A fraudulent transaction correctly identified as fraudulent.
- **True Negative:** A legitimate transaction correctly identified as legitimate.
- **False Positive:** A legitimate transaction incorrectly identified as fraudulent.
- **False Negative:** A fraudulent transaction incorrectly identified as legitimate.

By varying the threshold for classifying a transaction as fraudulent, we can calculate TPR and FPR. Plotting these points creates the ROC curve. The area under this curve is the AUROC.

Interpretation

- **AUROC ranges from 0 to 1.**
- **An AUROC of 1** indicates perfect classification.
- **An AUROC of 0.5** means the model is no better than random guessing.
- **Higher AUROC values indicate better performance.**

Advantages of AUROC

- **Independent of class distribution:** Works well with imbalanced datasets.
- **Considers all classification thresholds:** Provides a comprehensive view of model performance.

Limitations of AUROC

- **Doesn't directly optimize a specific metric:** Doesn't directly optimize precision, recall, or F1-score.

In essence, AUROC provides a valuable measure of a classification model's ability to distinguish between positive and negative classes.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

20. Mean Squared Error (MSE)

Mean Squared Error (MSE) is a metric used to evaluate the performance of a regression model. It measures the average squared difference between the predicted and actual values.

How it works

1. **Calculate the difference:** For each data point, subtract the predicted value from the actual value.
2. **Square the difference:** Square the difference calculated in step 1.
3. **Calculate the mean:** Find the average of all the squared differences.

Formula:

MSE = (1/n) * Σ(y_true - y_pred)^2

Where:

- n is the number of data points
- y_true is the actual value
- y_pred is the predicted value

Example

Let's say we're trying to predict house prices based on square footage. We have the following data:

Actual Price	Predicted Price
200,000	190,000
300,000	320,000

Calculate the differences:

- (200,000 - 190,000)^2 = 10,000
- (300,000 - 320,000)^2 = 40,000

Calculate the mean:

- MSE = (10,000 + 40,000) / 2 = 25,000

Interpretation

- **Lower MSE is better:** A lower MSE indicates that the model's predictions are closer to the actual values.
- **Sensitive to outliers:** Because MSE squares the errors, it gives more weight to larger errors, which can be sensitive to outliers.

In summary, MSE is a useful metric for evaluating regression models, but it's important to consider its sensitivity to outliers.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

21. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a metric used to measure the average magnitude of errors between predicted and actual values. Unlike Mean Squared Error (MSE), MAE doesn't square the differences, making it less sensitive to outliers.

How it works

1. **Calculate the absolute difference:** For each data point, find the absolute difference between the predicted and actual value.
2. **Calculate the mean:** Find the average of all the absolute differences.

Formula:

MAE = (1/n) * Σ|y_true - y_pred|

Where:

- n is the number of data points
- y_true is the actual value
- y_pred is the predicted value

Example

Let's use the same house price prediction example as before:

Actual Price	Predicted Price
200,000	190,000
300,000	320,000

Calculate the absolute differences:

- |200,000 - 190,000| = 10,000
- |300,000 - 320,000| = 20,000

Calculate the mean:

- MAE = (10,000 + 20,000) / 2 = 15,000

Interpretation

- **Lower MAE is better:** A lower MAE indicates that the model's predictions are closer to the actual values on average.
- **Less sensitive to outliers:** MAE is less affected by extreme values compared to MSE.

In summary, MAE provides a straightforward measure of prediction error and is often preferred when dealing with datasets that might contain outliers.

22. Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is a metric used to evaluate the performance of a regression model. It is the square root of the Mean Squared Error (MSE).

How it works

1. **Calculate the difference:** For each data point, subtract the predicted value from the actual value.
2. **Square the difference:** Square the difference calculated in step 1.
3. **Calculate the mean:** Find the average of all the squared differences (this is MSE).
4. **Take the square root:** Calculate the square root of the MSE.

Formula:

RMSE = sqrt((1/n) * Σ(y_true - y_pred)^2)

Where:

- n is the number of data points
- y_true is the actual value
- y_pred is the predicted value

Example

Let's use the same house price prediction example as before:

Actual Price	Predicted Price
200,000	190,000

300,000	320,000

Export to Sheets

We already calculated the MSE as 25,000.

- RMSE = sqrt(25,000) = 158.11

Interpretation

- **Lower RMSE is better:** A lower RMSE indicates that the model's predictions are closer to the actual values.
- **Same units as the data:** RMSE has the same units as the original data, making it easier to interpret.

In summary, RMSE is a popular metric for evaluating regression models as it provides a measure of the average magnitude of errors in the same units as the data.

23. Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is a metric used to assess the accuracy of a forecasting method by calculating the average absolute percentage error. It provides a relative measure of error, expressing accuracy as a ratio.

How it works

1. **Calculate the absolute percentage error:** For each data point, calculate the absolute difference between the predicted and actual value, divide it by the actual value, and then multiply by 100 to get a percentage.
2. **Calculate the mean:** Find the average of all the absolute percentage errors.

Formula:

MAPE = (1/n) * Σ(|y_true - y_pred| / y_true) * 100%

Where:

- n is the number of data points
- y_true is the actual value
- y_pred is the predicted value

Example

Let's use the same house price prediction example as before:

Actual Price	Predicted Price
200,000	190,000
300,000	320,000

Calculate the absolute percentage errors:

- |(200,000 - 190,000) / 200,000| * 100% = 5%
- |(300,000 - 320,000) / 300,000| * 100% = 6.67%

Calculate the mean:

- MAPE = (5% + 6.67%) / 2 = 5.83%

Interpretation

- **Lower MAPE is better:** A lower MAPE indicates that the model's predictions are closer to the actual values on average, expressed as a percentage.
- **Relatively easy to interpret:** MAPE provides a percentage-based measure of error, which is often easier to understand for non-technical audiences.
- **Sensitive to zero or near-zero values:** If actual values are close to zero, MAPE can be misleading or even undefined.

In summary, MAPE is a useful metric for evaluating forecast accuracy when dealing with positive values and when understanding the magnitude of errors in percentage terms is important.

24. Accuracy

Accuracy is a measure of how close a measured value is to the true or accepted value. It represents the correctness of a measurement.

Example

- **Temperature measurement:** If a thermometer reads 25 degrees Celsius when the actual temperature is 24.9 degrees Celsius, the thermometer is quite accurate.
- **Weight measurement:** If a scale shows a weight of 5 kilograms for a 5-kilogram object, the scale is accurate.

Calculation

Accuracy is often expressed as a percentage difference between the measured value and the true value:

Accuracy = [(Measured value - True value) / True value] * 100%

Important Considerations

- **Accuracy vs. Precision:** While accuracy refers to closeness to the true value, precision refers to the consistency or reproducibility of measurements.
- **Error:** Accuracy is often assessed in terms of error, which is the difference between the measured value and the true value.

A high level of accuracy is essential in many fields, including science, engineering, and medicine.

25. Precision

Precision refers to the closeness of repeated measurements to each other. It's about consistency and reproducibility, not necessarily how close the measurements are to the true value.

Example:

Imagine you're measuring the weight of an object five times.

- **Precise measurements:** 3.2 kg, 3.21 kg, 3.19 kg, 3.2 kg, 3.22 kg
- **Less precise measurements:** 2.8 kg, 3.5 kg, 4.1 kg, 2.9 kg, 3.3 kg

In the first example, the measurements are very close to each other, indicating high precision. In the second example, the measurements are spread out, indicating low precision.

Important to note:

- High precision doesn't necessarily mean high accuracy. Measurements can be precise but inaccurate if they consistently miss the true value.
- Precision is often measured by standard deviation.

26. Recall

Recall is a metric used to evaluate the performance of a classification model, particularly in information retrieval and machine learning. It measures the ability of a model to find all relevant instances.

Example:

Imagine a spam filter.

- **True Positives (TP):** Spam emails correctly identified as spam.
- **False Negatives (FN):** Spam emails incorrectly identified as not spam.

Recall is calculated as:

- Recall = TP / (TP + FN)

In essence, recall answers the question: "Of all the actual positive cases, how many did the model correctly identify?"

A high recall means that the model is good at finding all relevant instances, but it might also include some irrelevant ones (which is where precision comes into play).

27. HTER: Human-Targeted Translation Edit Rate

HTER (Human-Targeted Translation Edit Rate) is a metric used to evaluate the quality of machine translation (MT) output. It measures the amount of editing a human would have to perform to transform the machine-translated text into a fluent and accurate target language sentence.

How HTER Works

1. **Human Post-Editing:** A human editor modifies the machine-translated output to match the quality of a reference translation.
2. **Edit Calculation:** The number of edits required to transform the original machine translation into the human-edited version is counted.
3. **HTER Calculation:** The HTER is calculated as the ratio of the number of edits to the length of the reference translation.

HTER = Number of edits / Length of reference translation

Example

- **Machine Translation Output:** "The quick brown fox jumps over the lazy dog."
- **Reference Translation:** "The fast brown fox leaps over the sleepy dog."
- **Human-Edited Output:** "The fast brown fox jumps over the lazy dog."

In this example, the human editor would need to:

- Replace "quick" with "fast" (1 edit)
- Replace "jumps" with "leaps" (1 edit)
- Replace "lazy" with "sleepy" (1 edit)

Therefore, HTER = 3 edits / 7 words = 0.43

Key Differences from TER

- **TER** calculates the minimum number of edits required to match the machine translation to a reference translation.
- **HTER** focuses on the edits a human would actually make to improve the translation.

Advantages of HTER

- **Better correlation with human judgment:** HTER reflects the real-world effort required to post-edit machine translations.
- **Focuses on fluency and accuracy:** HTER considers both the semantic and linguistic quality of the translation.

In summary, HTER is a more human-centric metric for evaluating machine translation quality compared to traditional metrics like BLEU or TER. It provides a better understanding of the actual effort required to produce a high-quality translation.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

28. COMET: Crosslingual Optimized Metric for Evaluation of Translation

COMET (Crosslingual Optimized Metric for Evaluation of Translation) is a neural framework designed to predict human judgments of machine translation (MT) quality. Unlike traditional metrics like BLEU, which rely on n-gram overlap, COMET leverages the power of deep learning to capture more nuanced aspects of translation quality.

How COMET Works

1. **Model Training:** COMET is trained on a large dataset of human-rated translations. The model learns to predict human judgments of translation quality based on various features, including source and target language text, and potentially other auxiliary information.
2. **Evaluation:** Given a machine-translated output and a reference translation, COMET assigns a score that reflects the predicted human judgment of the translation's quality.

Key Features of COMET

- **Cross-lingual:** COMET can be applied to various language pairs, making it versatile for multilingual evaluation.
- **Human-aligned:** It is trained to predict human judgments, making it more correlated with human perception of quality.
- **Leverages deep learning:** COMET utilizes neural networks to capture complex language patterns and semantic relationships.

Example

Consider a machine translation of the sentence "The quick brown fox jumps over the lazy dog" from English to French. COMET would compare this machine translation to a human-produced reference translation and assign a score based on how closely the machine translation matches the quality of the human translation.

Advantages of COMET

- **Better correlation with human judgment:** COMET often outperforms traditional metrics in predicting human preferences.
- **Handles diverse language pairs:** It can be applied to a wide range of language combinations.
- **Captures semantic nuances:** COMET is able to capture deeper linguistic features than n-gram based metrics.

Limitations of COMET

- **Computational cost:** Training and using COMET can be computationally expensive.
- **Dependency on training data:** The quality of the model depends on the quality and quantity of the training data.

In summary, COMET represents a significant advancement in machine translation evaluation. By leveraging deep learning and focusing on human judgment, it provides a more accurate and informative assessment of translation quality compared to traditional metrics.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

29. chrF: Character n-gram F-score

chrF is a metric used to evaluate the quality of machine translation output. Unlike traditional metrics like BLEU which focus on word n-grams, chrF operates at the character level. This makes it particularly effective for languages with rich morphology, such as Finnish or Turkish.

How chrF works

1. **Character n-gram extraction:** Both the machine translation and reference translations are broken down into character n-grams (sequences of n characters).
2. **Matching:** The character n-grams from the machine translation are matched with those in the reference translation.
3. **Precision, Recall, and F-score:**
    - **Precision:** The ratio of matched character n-grams in the machine translation to the total number of character n-grams in the machine translation.
    - **Recall:** The ratio of matched character n-grams in the machine translation to the total number of character n-grams in the reference translation.
    - **F-score:** The harmonic mean of precision and recall, balancing the importance of both metrics.

Example

Consider the following:

- **Reference:** "the quick brown fox"
- **Machine Translation:** "the qvick brown fox"

For n-gram length of 3:

- Character n-grams in reference: "the", "he ", "e q", " qu", "qui", "uic", "ick", "ck ", "k b", " br", "bro", "row", "own", "wn ", "n f", " fo", "fox"
- Character n-grams in machine translation: "the", "he ", "e q", " qv", "qvi", "vic", "ick", "ck ", "k b", " br", "bro", "row", "own", "wn ", "n f", " fo", "fox"

By calculating the number of matching n-grams and applying the precision, recall, and F-score formulas, we can obtain the chrF score.

Advantages of chrF

- **Handles morphological variation:** Effective for languages with rich morphology.
- **Captures subword information:** Considers character-level patterns.
- **Correlates well with human judgment:** Often shows strong correlation with human evaluations.

Limitations of chrF

- **Might not capture all semantic nuances:** Focuses on character-level patterns rather than semantic meaning.
- **Sensitive to character-level errors:** Small character errors can significantly impact the score.

In summary, chrF is a valuable metric for evaluating machine translation, especially for languages with complex morphology. By considering character-level information, it provides a more nuanced assessment of translation quality compared to word-based metrics.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

30. SacreBLEU: Standardized BLEU

SacreBLEU is a Python library that provides a standardized implementation of the BLEU (Bilingual Evaluation Understudy) metric for evaluating machine translation quality. It addresses the inconsistencies and complexities often encountered when calculating BLEU scores.

Key Features of SacreBLEU

- **Standardization:** Ensures consistent BLEU score calculations across different implementations.
- **Tokenization:** Handles tokenization effectively, crucial for accurate BLEU computation.
- **Test Set Management:** Supports common test sets and their pre-processing.
- **Reproducibility:** Provides a clear specification of the BLEU calculation process.

How SacreBLEU Works

1. **Tokenization:** Text is tokenized using language-specific rules (e.g., mteval-v13a for English).
2. **N-gram Extraction:** N-grams (sequences of words) are extracted from both the reference and candidate translations.
3. **Precision Calculation:** Calculates the precision of n-grams in the candidate translation compared to the reference translation.
4. **Length Penalty:** Adjusts the score based on the length difference between the candidate and reference translations.
5. **BLEU Score Calculation:** Combines the precision and length penalty to compute the final BLEU score.

Example

Let's say you have a machine translation output and its corresponding reference translation. You can use the SacreBLEU library to calculate the BLEU score:

```
import sacrebleu
references = [["the quick brown fox jumps over the lazy dog"]]
candidate = "the quick brown fox jumped over the lazy dog"
score = sacrebleu.sentence_bleu(references, [candidate])
print(score)
```

Output:

```
BLEU = 0.9500 (13.0/13.0/13.0/13.0)
```

Advantages of SacreBLEU

- Ensures consistent BLEU scores across different systems and implementations.
- Simplifies the BLEU calculation process.
- Provides standardized tokenization and test set handling.

By using SacreBLEU, researchers and practitioners can more reliably compare machine translation systems and track performance improvements over time.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

1. Language Model Evaluation

Metric	Description
Perplexity (PPL)	Measures the perplexity of a language model.
BLEU	Bilingual Evaluation Understudy for machine translation.
ROUGE	Recall-Oriented Understudy for Gisting Evaluation.
METEOR	Metric for Evaluation of Translation with Explicit Ordering.
TER	Translation Edit Rate.
GLEU	Google BLEU.
SacreBLEU	Standardized BLEU.
chrF	Character n-gram F-score.
COMET	Crosslingual Optimized Metric for Evaluation of Translation.
HTER	Human-Targeted Translation Edit Rate.
BERTScore	BERT-based Scoring.
BLEURT	BERT-based Language Understanding Evaluation Reference Tool.
MAUVE	Measuring the Gap between Neural Text and Human Text.

2. Classification and Regression Evaluation

Metric	Description
Accuracy	Proportion of correct predictions.
Precision	Proportion of positive predictions that are truly positive.
Recall	Proportion of actual positives that are correctly identified.
F1 Score	Harmonic mean of precision and recall.
AUC-ROC	Area Under the Receiver Operating Characteristic curve.
AUROC	Area Under the Receiver Operating Characteristic curve.
MSE	Mean Squared Error.
MAE	Mean Absolute Error.
RMSE	Root Mean Square Error.
MAPE	Mean Absolute Percentage Error.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g

3. Information Retrieval Evaluation

Metric	Description
MRR	Mean Reciprocal Rank.
NDCG	Normalized Discounted Cumulative Gain.
MAP	Mean Average Precision.

4. Image Description Evaluation

Metric	Description
CIDEr	Consensus-based Image Description Evaluation.
SPICE	Semantic Propositional Image Caption Evaluation.

Note: Some metrics can be used in multiple contexts. For example, BLEU can be used for both machine translation and text summarization.

Created by : Syed Afroz Ali
Follow me on LinkedIn for insights and updates: https://lnkd.in/gxcsx77g