Import and Install all Liberaries

```
%%capture
! pip install torch==2.6.0+cpu
!pip install transformers==4.46.3
# !python --version

from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import (
    mean_absolute_error,
    mean_squared_error,
    r2_score,
    mean_absolute_percentage_error,
    median_absolute_error
)
import matplotlib.pyplot as plt
from matplotlib.colors import rgb2hex
from scipy.stats import spearmanr, kendalltau, pearsonr
import json
from tqdm import tqdm
```

Evaluate the Score

```
data_path = "/content/dataset.json"

with open(data_path, "r") as f:
    dataset = json.load(f)

print(dataset[0]["pr_code"])

class Car:
    def __init__(self, make, model):
        self.make = make
        self.model = model

    def start(self):
        print(f'{self.make} {self.model} starts!')

# Example usage:
car = Car("Toyota", "Corolla")
car.start()


hf_token = "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"
```

```python
# add token
from huggingface_hub import notebook_login
notebook_login(hf_token)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_deprecation.py:38: FutureWarning: Deprecated positional argument(s)
used in 'notebook_login': pass
new_session='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' as keyword args.
From version 1.0 passing these as positional arguments will result in
an error,
  warnings.warn(
```

{"model_id":"acb82cd8e6c9467d870a1a0f2c186c02","version_major":2,"vers
ion_minor":0}

````python
from huggingface_hub import InferenceClient
def score_code(prof_code,stud_code,model):
  client =
InferenceClient(api_key="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")#hf_HL
dGCUqbIWzgJOGaLUwacdwIOHgKYDxdrl

  messages = [
    {
        "role": "user",
        "content": f"""
Compare la **logique** des deux codes Python ci-dessous et retourne
**exclusivement** un score de similarité entre 0 et 100%.

□ **Critères de comparaison** :
- Compare **uniquement la logique et l’algorithme** (ignore les noms
de variables, les commentaires, l’ordre des définitions si l’exécution
est équivalente).
- Utilise **une analyse structurelle et algorithmique approfondie**
(ne te base pas uniquement sur les mots-clés ou la syntaxe).
- **100%** = Les codes sont **logiquement identiques** ou quasi-
identiques.
- **0%** = Les codes sont **totalement différents** en termes de
logique.
- **Les valeurs intermédiaires (ex. 70%, 50%, 30%) sont autorisées**
et doivent refléter le degré de similitude.

□ **Restrictions obligatoires** :
- **NE DONNE AUCUNE EXPLICATION.**
- **NE FOURNIS AUCUN TEXTE SUPPLÉMENTAIRE avant ou après le score.**
- **Si tu ne peux pas respecter ce format, ne retourne rien.**

□ **Codes à comparer** :

Code du professeur :
```python
````
````

```
{prof_code}
```

Code de l'étudiant :
```python
        {stud_code}```
Retourne uniquement le score sous ce format :
similarity score: X%
(où X est une valeur entre 0 et 100) """ } ]


  import re

  completion = client.chat.completions.create(
    model=model,
    messages=messages,
    max_tokens=100
    )

  def find_score(content):
      scores = re.findall(r"(\d+\.\d+|\d+)", content)  # Match both
entiers et décimaux
      for score in scores:
          score = float(score)
          if 0 <= score <= 100:
              return score  # Retourne dès qu'on trouve un score
valide
      return 0  # Si aucun score valide trouvé, retourne 0

  # Récupération de la réponse
  text = getattr(completion, 'choices', [{}])[0].get('message',
{}).get('content', "")

  if text:
      return find_score(text)
  return 0  # Si le texte est vide, retourne 0

from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
import numpy as np
import pandas as pd
import json
import matplotlib.pyplot as plt
from tqdm import tqdm
from sklearn.metrics import (
    mean_absolute_error,
    mean_squared_error,
    r2_score,
    mean_absolute_percentage_error,
    median_absolute_error
```

```
)
from scipy.stats import spearmanr, kendalltau, pearsonr

# Model Checkpoints to Compare
MODEL_CHECKPOINTS = [
    "meta-llama/Meta-Llama-3-8B-Instruct",#
    "google/gemma-2-9b-it",#
    "google/gemma-2-27b-it",#
    "HuggingFaceTB/SmolLM2-1.7B-Instruct",
    "meta-llama/Llama-3.2-1B-Instruct",
    "HuggingFaceTB/SmolLM2-1.7B-Instruct",
    "mistralai/Mistral-Nemo-Instruct-2407",
    "microsoft/Phi-3-mini-4k-instruct",
    "google/gemma-2-2b-it",
    "mistralai/Mixtral-8x7B-Instruct-v0.1",

]

DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load dataset
data_path = "dataset.json"
with open(data_path, "r") as f:
    dataset = json.load(f)
#def score_code(pr_code, st_code, model_checkpoint):
    """
    Dummy function for model inference - replace with actual scoring
logic.
    """
    # Replace this with actual model inference (e.g., via Hugging
Face's pipeline)
 #   return np.random.uniform(0, 100)

def evaluate_model(model_checkpoint):
    """
    Runs inference using the given model checkpoint and evaluates
performance.
    """
    tr_scores = []
    pr_scores = []

    for entry in tqdm(dataset, desc=f"Evaluating {model_checkpoint}"):
        pr_code = entry.get("pr_code")
        st_code = entry.get("st_code")
        tr_score = entry.get("score")

        tr_scores.append(tr_score)

        pr_score = score_code(pr_code, st_code, model_checkpoint)
```

```
        if pr_score is not None:
            pr_scores.append(pr_score)
        else:
            print(f"Warning: No valid score for {pr_code[:30]}... and
{st_code[:30]}...")

    return tr_scores, pr_scores

def huber_loss_custom(y_true, y_pred, delta=1.0):
    error = y_true - y_pred
    abs_error = np.abs(error)
    quadratic = np.minimum(abs_error, delta)
    linear = abs_error - quadratic
    return np.mean(0.5 * quadratic**2 + delta * linear)

def get_cell_color(metric_name, value):
    lower_is_better_keywords = [
        "error", "loss", "mae", "mse", "rmse", "mape", "rae", "rse",
"quantile", "median"
    ]
    lower_is_better = any(kw in metric_name.lower() for kw in
lower_is_better_keywords)

    if 0 <= value <= 1:
        perc = value * 100
        display_str = f"{perc:.2f}%"
    else:
        perc = value
        display_str = f"{value:.2f}"

    discrete_val = round(perc / 10) * 10
    norm = discrete_val / 100.0
    cmap = plt.get_cmap('RdYlGn')

    if not lower_is_better:
        color = cmap(norm)
    else:
        color = cmap(1 - norm)

    color_hex = rgb2hex(color)

    return color_hex, display_str

def evaluate_regression_model(y_true, y_pred, quantile=0.9):
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    # Compute Metrics
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
```

```
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    mape = mean_absolute_percentage_error(y_true, y_pred)
    medae = median_absolute_error(y_true, y_pred)

    # Relative Errors
    baseline_mae = np.sum(np.abs(y_true - np.mean(y_true)))
    rae = np.sum(np.abs(y_true - y_pred)) / baseline_mae if
baseline_mae != 0 else np.nan
    baseline_mse = np.sum((y_true - np.mean(y_true))**2)
    rse = np.sum((y_true - y_pred)**2) / baseline_mse if baseline_mse
!= 0 else np.nan

    # Huber Loss
    huber = huber_loss_custom(y_true, y_pred)

    # Quantile Loss
    diff = y_true - y_pred
    quantile_loss = np.mean(np.maximum(quantile * diff, (quantile - 1)
* diff))

    # Ranking/Correlation Metrics
    spearman_corr, _ = spearmanr(y_true, y_pred)
    kendall_corr, _ = kendalltau(y_true, y_pred)
    pearson_corr, _ = pearsonr(y_true, y_pred)

    # All metrics
    metrics_dict = {
        "Mean Absolute Error (MAE)": mae,
        "Mean Squared Error (MSE)": mse,
        "Root Mean Squared Error (RMSE)": rmse,
        "R-squared (R²)": r2,
        "Mean Absolute Percentage Error (MAPE)": mape,
        "Median Absolute Error": medae,
        "Relative Absolute Error (RAE)": rae,
        "Relative Squared Error (RSE)": rse,
        "Huber Loss": huber,
        f"Quantile Loss (q={quantile})": quantile_loss,
        "Spearman’s Rank Correlation": spearman_corr,
        "Kendall’s Tau": kendall_corr,
        "Pearson’s Correlation": pearson_corr,
    }

    # Metrics to DataFrame
    metrics_df = pd.DataFrame(list(metrics_dict.items()),
columns=["Metric", "Value"])

    # ------------------------------------------------------
    #                    Table of Metrics                  #
    # ------------------------------------------------------
```

```
    fig_table, ax_table = plt.subplots(figsize=(6, len(metrics_df) *
0.3 + 1))
    ax_table.axis('off')

    table = ax_table.table(
        cellText=metrics_df.values,
        colLabels=metrics_df.columns,
        cellLoc='center',
        loc='center'
    )
    table.auto_set_font_size(False)
    table.set_fontsize(10)
    table.scale(1, 1.5)

    for (row, col), cell in table.get_celld().items():
        if row == 0:
            cell.set_facecolor("#f0f0f0")
            continue
        if col == 1:
            metric_name = metrics_df.iloc[row - 1]["Metric"]
            metric_value = metrics_df.iloc[row - 1]["Value"]
            color, display_str = get_cell_color(metric_name,
metric_value)
            cell.get_text().set_text(display_str)
            cell.set_facecolor(color)

    plt.title("Regression Model Evaluation Metrics", fontsize=14,
pad=20)
    plt.tight_layout()
    plt.show()

    # ------------------------------------------------------
    #       Visualizations for Model Diagnostics           #
    # ------------------------------------------------------
    fig, axs = plt.subplots(1, 3, figsize=(18, 5))
    axs[0].scatter(y_true, y_pred, alpha=0.7, label='Data points')
    min_val = min(np.min(y_true), np.min(y_pred))
    max_val = max(np.max(y_true), np.max(y_pred))
    axs[0].plot([min_val, max_val], [min_val, max_val], 'r--',
label='Ideal')
    axs[0].set_xlabel('Actual Values')
    axs[0].set_ylabel('Predicted Values')
    axs[0].set_title('Predicted vs. Actual')
    axs[0].legend()

    # Histogram of Residuals
    residuals = y_true - y_pred
    axs[1].hist(residuals, bins=20, edgecolor='k', alpha=0.7)
    axs[1].set_title('Residuals Histogram')
    axs[1].set_xlabel('Residual')
```

```
    axs[1].set_ylabel('Frequency')

    # Line Plot
    indices = np.arange(len(y_true))
    axs[2].plot(indices, y_true, label='Actual', marker='o',
linestyle='-')
    axs[2].plot(indices, y_pred, label='Predicted', marker='x',
linestyle='--')
    axs[2].set_title('Actual vs. Predicted Over Index')
    axs[2].set_xlabel('Index')
    axs[2].set_ylabel('Value')
    axs[2].legend()

    plt.tight_layout()
    plt.show()

    return metrics_dict

# Run evaluation for all models
results = {}

for model in MODEL_CHECKPOINTS:
    tr_scores, pr_scores = evaluate_model(model)
    metrics = evaluate_regression_model(tr_scores, pr_scores)
    results[model] = metrics

# Convert results to DataFrame for better visualization
df_results = pd.DataFrame(results).T

# Print results
print(df_results)

# Plot results
df_results.plot(kind="bar", figsize=(12, 6), colormap="viridis")
plt.title("Model Performance Comparison")
plt.ylabel("Metric Value")
plt.xticks(rotation=45, ha="right")
plt.grid(axis="y")
plt.legend(loc="best")
plt.show()

# Réorganiser le DataFrame pour mieux afficher les résultats sous
forme de tableau
df_results = df_results.reset_index().rename(columns={"index":
"Model"})

# --------------------------------------------------------
#                    Table of Metrics
# --------------------------------------------------------
fig_table, ax_table = plt.subplots(figsize=(16, len(df_results) * 1.5
```

```
+ 2))  # Increased figure size for better visual
ax_table.axis('off')

table = ax_table.table(
    cellText=df_results.values,
    colLabels=df_results.columns,
    cellLoc='center',
    loc='center'
)

table.auto_set_font_size(False)
table.set_fontsize(14)  # Increase font size for better readability
table.scale(2.5, 2.5)  # Increase row height and column width for
larger table

# Adjust font sizes for header and first column
for (row, col), cell in table.get_celld().items():
    if row == 0:  # Header row
        cell.set_facecolor("#f0f0f0")
        cell.set_fontsize(12)  # Smaller font for header
        cell.set_text_props(weight="bold")
        cell.set_height(0.1)  # Increase row height for header
        continue
    if col == 0:  # First column (Metric names)
        cell.set_fontsize(12)  # Slightly larger font for metric names
        cell.set_width(0.25)  # Increase column width for metric names
        continue
    if col > 0:  # Value columns
        metric_name = df_results.columns[col]
        metric_value = df_results.iloc[row - 1, col]
        color, display_str = get_cell_color(metric_name, metric_value)
        cell.get_text().set_text(display_str)
        cell.set_facecolor(color)
        cell.set_width(0.2)  # Adjust width of value columns
        cell.set_height(0.1)  # Increase row height for value columns

plt.title("Comparison of Regression Model Evaluation Metrics",
fontsize=18, pad=20)  # Bigger title
plt.tight_layout()
plt.show()

# Prepare the figure and axis
fig, ax = plt.subplots(figsize=(16, len(df_results) * 1.5 + 2))

# Plot for each model
for model, metrics in results.items():
    # Get the true and predicted values from the evaluation function
    tr_scores, pr_scores = evaluate_model(model)  # Get the true and
predicted scores
```

```
    # Generate indices for x-axis
    indices = np.arange(len(tr_scores))

    # Plot actual vs predicted for each model

    ax.plot(indices, pr_scores, label=f'{model} - Predicted',
marker='x', linestyle='--', alpha=0.7)

ax.plot(indices, tr_scores, label='Actual', marker='o', linestyle='-',
alpha=0.7)
# Set the title and labels
ax.set_title('Actual vs. Predicted Over Index for All Models',
fontsize=14)
ax.set_xlabel('Index', fontsize=12)
ax.set_ylabel('Value', fontsize=12)

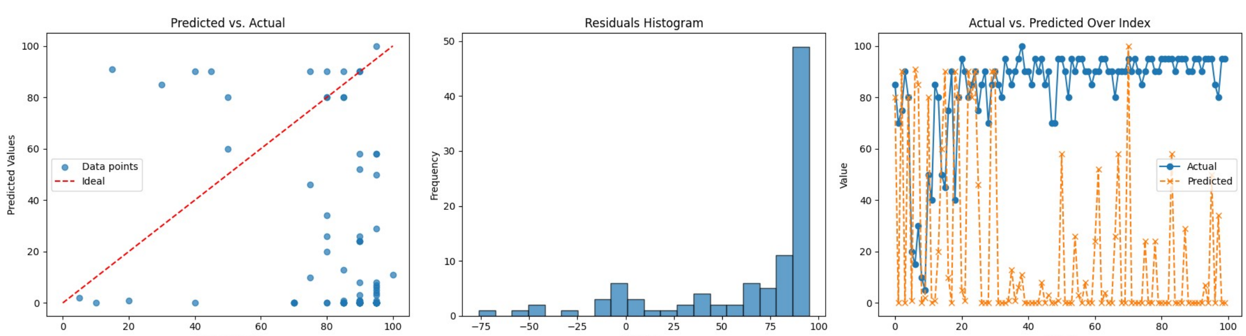
# Add a legend to differentiate between the models
ax.legend(loc='best', fontsize=10)

# Display the plot with tight layout
plt.tight_layout()
plt.show()
```

```
Evaluating meta-llama/Meta-Llama-3-8B-Instruct: 100%|██████████|
100/100 [00:07<00:00, 14.24it/s]
```

Regression Model Evaluation Metrics

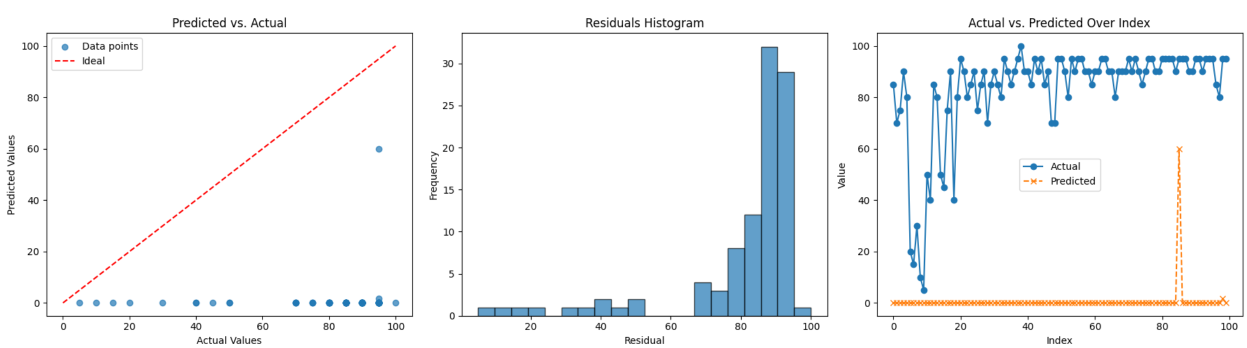
Metric	Value
Mean Absolute Error (MAE)	69.21
Mean Squared Error (MSE)	5737.27
Root Mean Squared Error (RMSE)	75.74
R-squared (R^2)	-14.38
Mean Absolute Percentage Error (MAPE)	87.16%
Median Absolute Error	85.00
Relative Absolute Error (RAE)	5.54
Relative Squared Error (RSE)	15.38
Huber Loss	68.73
Quantile Loss (q=0.9)	59.88
Spearman’s Rank Correlation	-0.24
Kendall’s Tau	-0.20
Pearson’s Correlation	-0.28




```
Evaluating google/gemma-2-9b-it: 100%|██████████| 100/100
[00:07<00:00, 12.94it/s]
```

Regression Model Evaluation Metrics

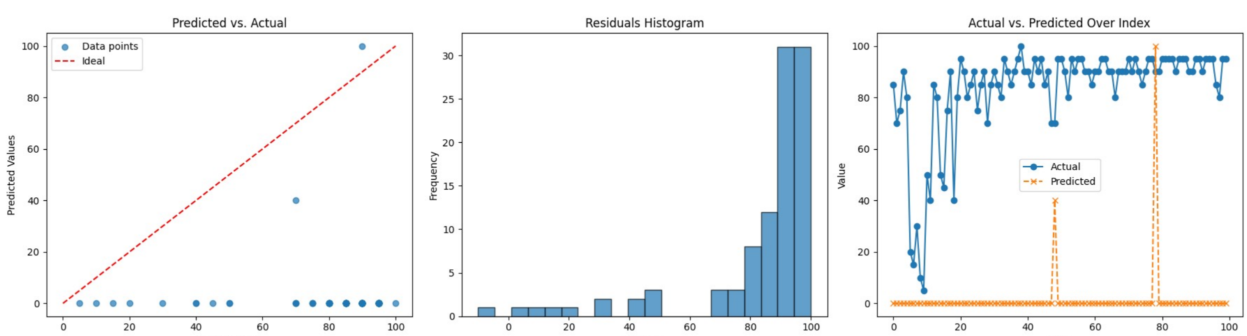
Metric	Value
Mean Absolute Error (MAE)	82.38
Mean Squared Error (MSE)	7180.99
Root Mean Squared Error (RMSE)	84.74
R-squared (R²)	-18.25
Mean Absolute Percentage Error (MAPE)	99.35%
Median Absolute Error	90.00
Relative Absolute Error (RAE)	6.59
Relative Squared Error (RSE)	19.25
Huber Loss	81.88
Quantile Loss (q=0.9)	74.15
Spearman’s Rank Correlation	17.37%
Kendall’s Tau	15.49%
Pearson’s Correlation	6.41%




```
Evaluating google/gemma-2-27b-it: 100%|██████████| 100/100
[00:06<00:00, 15.36it/s]
```

Regression Model Evaluation Metrics

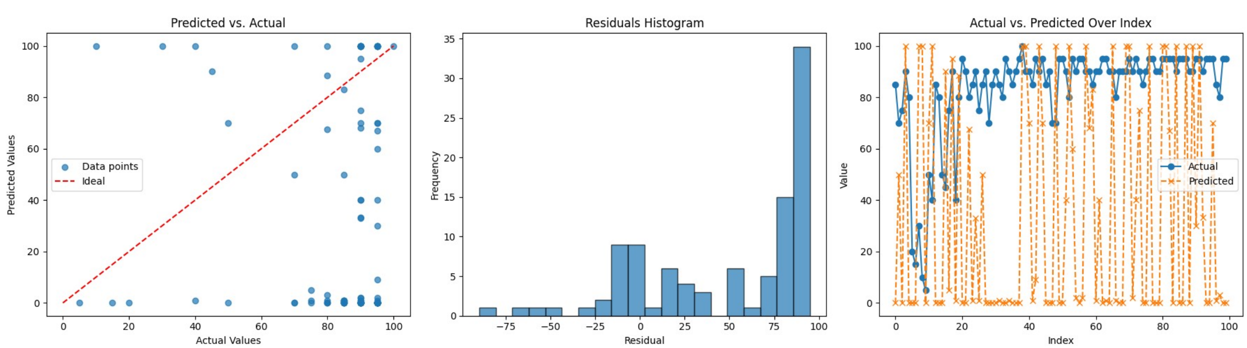
Metric	Value
Mean Absolute Error (MAE)	81.80
Mean Squared Error (MSE)	7142.00
Root Mean Squared Error (RMSE)	84.51
R-squared (R^2)	-18.15
Mean Absolute Percentage Error (MAPE)	98.54%
Median Absolute Error	90.00
Relative Absolute Error (RAE)	6.54
Relative Squared Error (RSE)	19.15
Huber Loss	81.30
Quantile Loss (q=0.9)	73.54
Spearman’s Rank Correlation	-0.09
Kendall’s Tau	-0.08
Pearson’s Correlation	0.87%




```
Evaluating HuggingFaceTB/SmolLM2-1.7B-Instruct: 100%|██████████|
100/100 [00:07<00:00, 14.20it/s]
```

Regression Model Evaluation Metrics

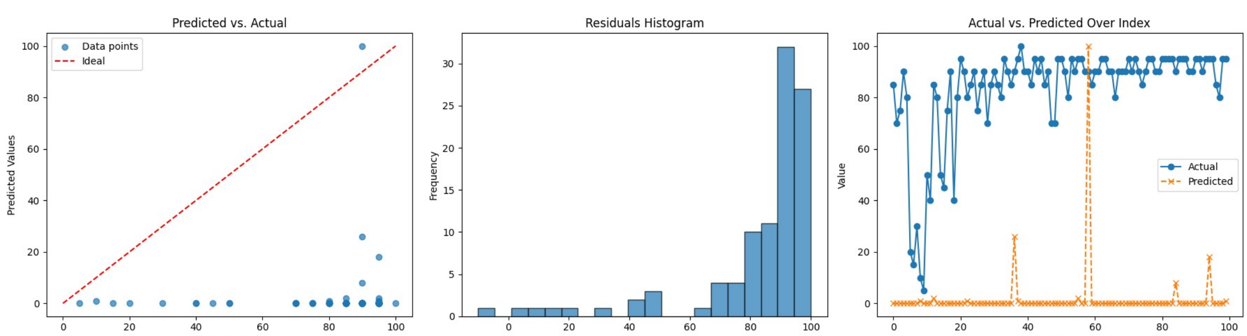
Metric	Value
Mean Absolute Error (MAE)	58.94
Mean Squared Error (MSE)	4684.36
Root Mean Squared Error (RMSE)	68.44
R-squared (R²)	-11.56
Mean Absolute Percentage Error (MAPE)	81.22%
Median Absolute Error	76.00
Relative Absolute Error (RAE)	4.71
Relative Squared Error (RSE)	12.56
Huber Loss	58.44
Quantile Loss (q=0.9)	49.37
Spearman’s Rank Correlation	1.12%
Kendall’s Tau	0.73%
Pearson’s Correlation	-0.04




```
Evaluating meta-llama/Llama-3.2-1B-Instruct: 100%|██████████| 100/100
[00:07<00:00, 13.46it/s]
```

Regression Model Evaluation Metrics

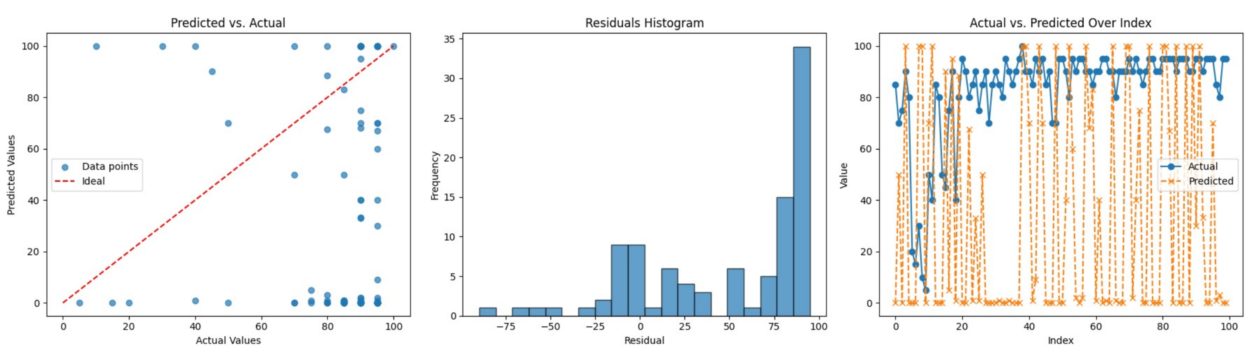
Metric	Value
Mean Absolute Error (MAE)	81.60
Mean Squared Error (MSE)	7084.56
Root Mean Squared Error (RMSE)	84.17
R-squared (R^2)	-17.99
Mean Absolute Percentage Error (MAPE)	98.37%
Median Absolute Error	90.00
Relative Absolute Error (RAE)	6.53
Relative Squared Error (RSE)	18.99
Huber Loss	81.10
Quantile Loss (q=0.9)	73.36
Spearman's Rank Correlation	6.11%
Kendall's Tau	5.27%
Pearson's Correlation	5.62%




```
Evaluating HuggingFaceTB/SmolLM2-1.7B-Instruct: 100%|██████████|
100/100 [00:07<00:00, 13.35it/s]
```

Regression Model Evaluation Metrics

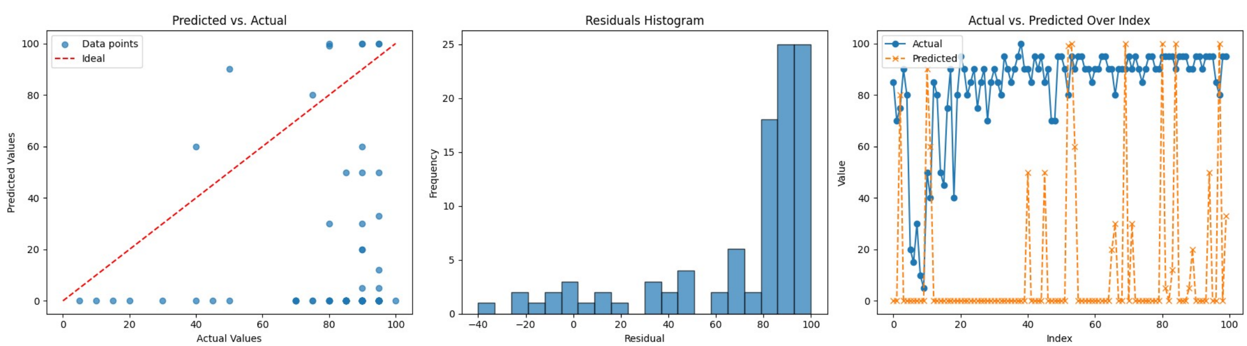
Metric	Value
Mean Absolute Error (MAE)	58.94
Mean Squared Error (MSE)	4684.36
Root Mean Squared Error (RMSE)	68.44
R-squared (R²)	-11.56
Mean Absolute Percentage Error (MAPE)	81.22%
Median Absolute Error	76.00
Relative Absolute Error (RAE)	4.71
Relative Squared Error (RSE)	12.56
Huber Loss	58.44
Quantile Loss (q=0.9)	49.37
Spearman's Rank Correlation	1.12%
Kendall's Tau	0.73%
Pearson's Correlation	-0.04




```
Evaluating mistralai/Mistral-Nemo-Instruct-2407: 100%|██████████|
100/100 [00:08<00:00, 11.71it/s]
```

Regression Model Evaluation Metrics

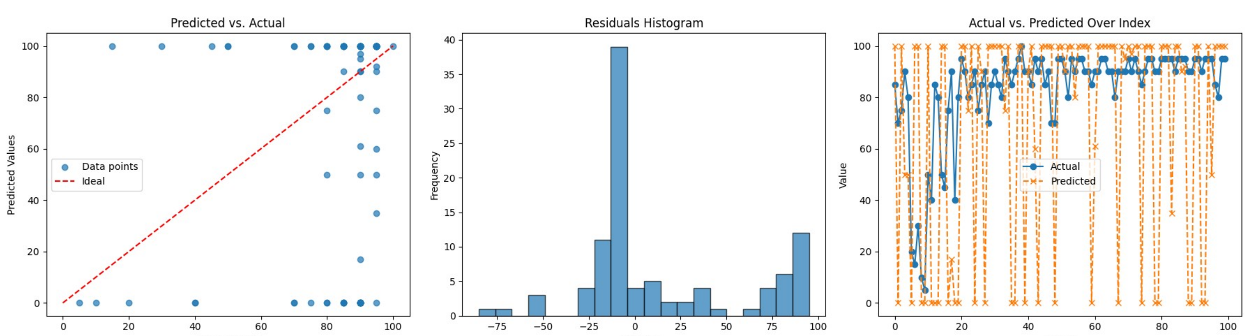
Metric	Value
Mean Absolute Error (MAE)	73.74
Mean Squared Error (MSE)	6218.44
Root Mean Squared Error (RMSE)	78.86
R-squared (R²)	-15.67
Mean Absolute Percentage Error (MAPE)	89.11%
Median Absolute Error	87.50
Relative Absolute Error (RAE)	5.90
Relative Squared Error (RSE)	16.67
Huber Loss	73.24
Quantile Loss (q=0.9)	65.29
Spearman's Rank Correlation	-0.01
Kendall's Tau	-0.01
Pearson's Correlation	-0.01




```
Evaluating microsoft/Phi-3-mini-4k-instruct: 100%|██████████| 100/100
[00:07<00:00, 13.00it/s]
```

Regression Model Evaluation Metrics

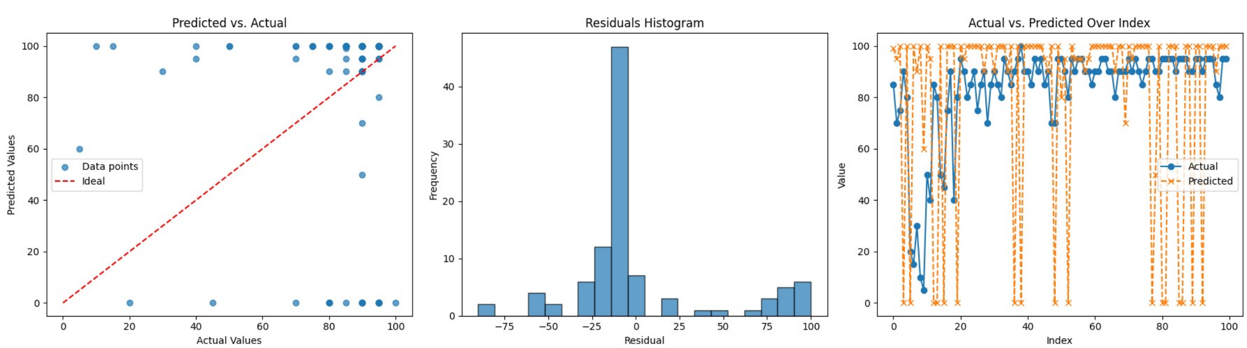
Metric	Value
Mean Absolute Error (MAE)	31.37
Mean Squared Error (MSE)	2034.03
Root Mean Squared Error (RMSE)	45.10
R-squared (R²)	-4.45
Mean Absolute Percentage Error (MAPE)	47.72%
Median Absolute Error	15.00
Relative Absolute Error (RAE)	2.51
Relative Squared Error (RSE)	5.45
Huber Loss	30.89
Quantile Loss (q=0.9)	21.30
Spearman's Rank Correlation	25.41%
Kendall's Tau	21.79%
Pearson's Correlation	24.61%




```
Evaluating google/gemma-2-2b-it: 100%|██████████| 100/100
[00:08<00:00, 12.25it/s]
```

Regression Model Evaluation Metrics

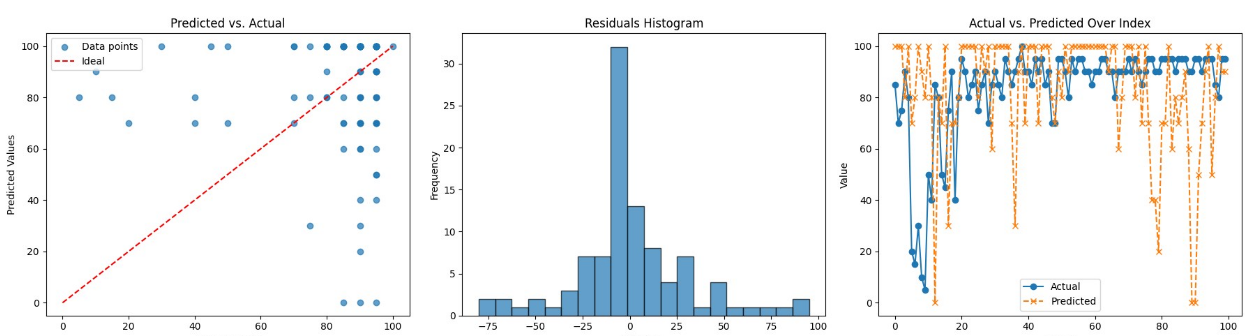
Metric	Value
Mean Absolute Error (MAE)	26.69
Mean Squared Error (MSE)	1674.71
Root Mean Squared Error (RMSE)	40.92
R-squared (R²)	-3.49
Mean Absolute Percentage Error (MAPE)	58.67%
Median Absolute Error	10.00
Relative Absolute Error (RAE)	2.14
Relative Squared Error (RSE)	4.49
Huber Loss	26.23
Quantile Loss (q=0.9)	14.43
Spearman's Rank Correlation	7.60%
Kendall's Tau	6.42%
Pearson's Correlation	5.70%




```
Evaluating mistralai/Mixtral-8x7B-Instruct-v0.1: 100%|██████████|
100/100 [00:13<00:00,  7.51it/s]
```

Regression Model Evaluation Metrics

Metric	Value
Mean Absolute Error (MAE)	21.60
Mean Squared Error (MSE)	941.00
Root Mean Squared Error (RMSE)	30.68
R-squared (R^2)	-1.52
Mean Absolute Percentage Error (MAPE)	55.00%
Median Absolute Error	15.00
Relative Absolute Error (RAE)	1.73
Relative Squared Error (RSE)	2.52
Huber Loss	21.13
Quantile Loss (q=0.9)	10.80
Spearman's Rank Correlation	-0.01
Kendall's Tau	-0.01
Pearson's Correlation	-0.02




```
                                    Mean Absolute Error (MAE)  \
meta-llama/Meta-Llama-3-8B-Instruct                    69.2100
google/gemma-2-9b-it                                   82.3840
google/gemma-2-27b-it                                  81.8000
HuggingFaceTB/SmolLM2-1.7B-Instruct                    58.9357
meta-llama/Llama-3.2-1B-Instruct                       81.6000
mistralai/Mistral-Nemo-Instruct-2407                   73.7400
microsoft/Phi-3-mini-4k-instruct                       31.3700
google/gemma-2-2b-it                                   26.6900
mistralai/Mixtral-8x7B-Instruct-v0.1                   21.6000
```

```
                                      Mean Squared Error (MSE)  \
meta-llama/Meta-Llama-3-8B-Instruct                5737.270000
google/gemma-2-9b-it                               7180.985600
google/gemma-2-27b-it                              7142.000000
HuggingFaceTB/SmolLM2-1.7B-Instruct                4684.362989
meta-llama/Llama-3.2-1B-Instruct                   7084.560000
mistralai/Mistral-Nemo-Instruct-2407               6218.440000
microsoft/Phi-3-mini-4k-instruct                   2034.030000
google/gemma-2-2b-it                               1674.710000
mistralai/Mixtral-8x7B-Instruct-v0.1                941.000000

                                      Root Mean Squared Error
(RMSE)  \
meta-llama/Meta-Llama-3-8B-Instruct                              75.744769

google/gemma-2-9b-it                                             84.740696

google/gemma-2-27b-it                                            84.510354

HuggingFaceTB/SmolLM2-1.7B-Instruct                              68.442406

meta-llama/Llama-3.2-1B-Instruct                                 84.169828

mistralai/Mistral-Nemo-Instruct-2407                             78.857086

microsoft/Phi-3-mini-4k-instruct                                 45.100222

google/gemma-2-2b-it                                             40.923221

mistralai/Mixtral-8x7B-Instruct-v0.1                             30.675723


                                      R-squared (R²)  \
meta-llama/Meta-Llama-3-8B-Instruct       -14.381421
google/gemma-2-9b-it                      -18.251972
google/gemma-2-27b-it                     -18.147453
HuggingFaceTB/SmolLM2-1.7B-Instruct       -11.558614
meta-llama/Llama-3.2-1B-Instruct          -17.993458
mistralai/Mistral-Nemo-Instruct-2407      -15.671421
microsoft/Phi-3-mini-4k-instruct           -4.453164
google/gemma-2-2b-it                       -3.489839
mistralai/Mixtral-8x7B-Instruct-v0.1       -1.522788

                                      Mean Absolute Percentage Error
(MAPE)  \
meta-llama/Meta-Llama-3-8B-Instruct
0.871555
google/gemma-2-9b-it
0.993516
google/gemma-2-27b-it
```

```
0.985397
HuggingFaceTB/SmolLM2-1.7B-Instruct
0.812194
meta-llama/Llama-3.2-1B-Instruct
0.983657
mistralai/Mistral-Nemo-Instruct-2407
0.891102
microsoft/Phi-3-mini-4k-instruct
0.477233
google/gemma-2-2b-it
0.586719
mistralai/Mixtral-8x7B-Instruct-v0.1
0.550013

                                      Median Absolute Error  \
meta-llama/Meta-Llama-3-8B-Instruct                    85.0
google/gemma-2-9b-it                                   90.0
google/gemma-2-27b-it                                  90.0
HuggingFaceTB/SmolLM2-1.7B-Instruct                    76.0
meta-llama/Llama-3.2-1B-Instruct                       90.0
mistralai/Mistral-Nemo-Instruct-2407                   87.5
microsoft/Phi-3-mini-4k-instruct                       15.0
google/gemma-2-2b-it                                   10.0
mistralai/Mixtral-8x7B-Instruct-v0.1                   15.0

                                      Relative Absolute Error (RAE)  \
meta-llama/Meta-Llama-3-8B-Instruct                        5.536800
google/gemma-2-9b-it                                       6.590720
google/gemma-2-27b-it                                      6.544000
HuggingFaceTB/SmolLM2-1.7B-Instruct                        4.714856
meta-llama/Llama-3.2-1B-Instruct                           6.528000
mistralai/Mistral-Nemo-Instruct-2407                       5.899200
microsoft/Phi-3-mini-4k-instruct                           2.509600
google/gemma-2-2b-it                                       2.135200
mistralai/Mixtral-8x7B-Instruct-v0.1                       1.728000

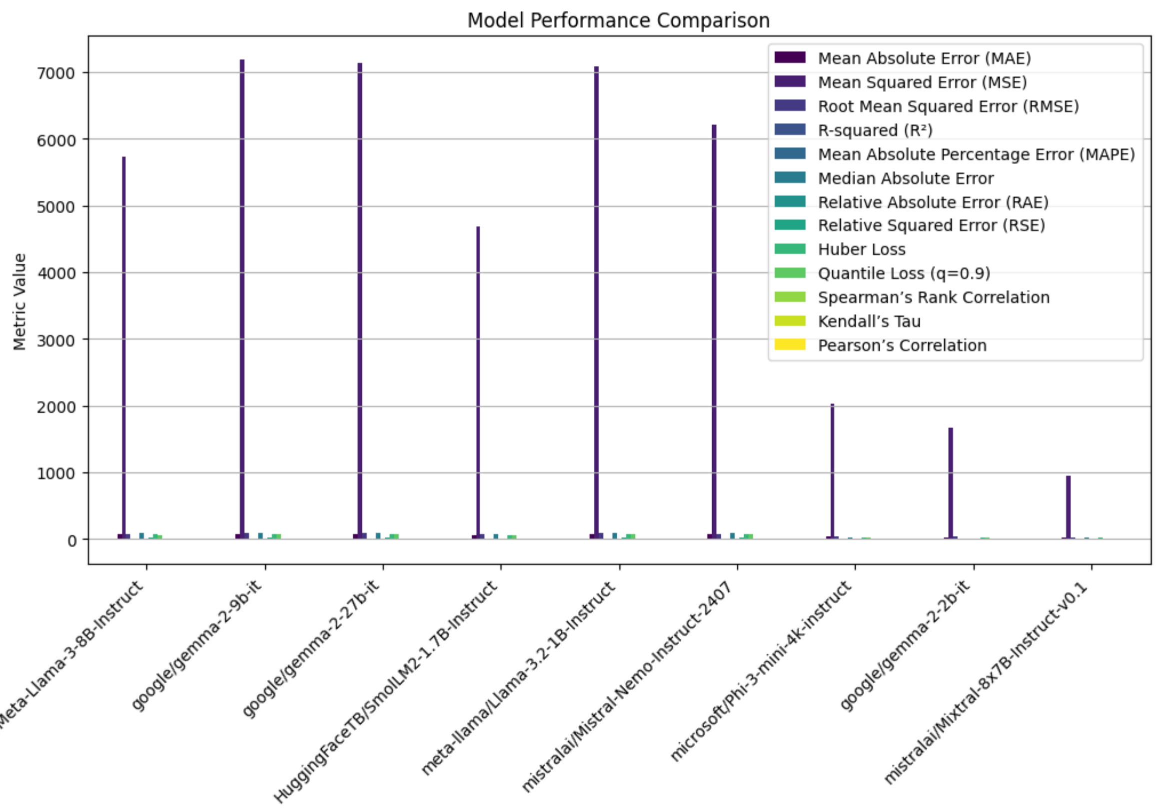
                                      Relative Squared Error (RSE)  \
meta-llama/Meta-Llama-3-8B-Instruct                      15.381421
google/gemma-2-9b-it                                     19.251972
google/gemma-2-27b-it                                    19.147453
HuggingFaceTB/SmolLM2-1.7B-Instruct                      12.558614
meta-llama/Llama-3.2-1B-Instruct                         18.993458
mistralai/Mistral-Nemo-Instruct-2407                     16.671421
microsoft/Phi-3-mini-4k-instruct                          5.453164
google/gemma-2-2b-it                                      4.489839
mistralai/Mixtral-8x7B-Instruct-v0.1                      2.522788

                                      Huber Loss  Quantile Loss
(q=0.9)  \
meta-llama/Meta-Llama-3-8B-Instruct      68.7300
```

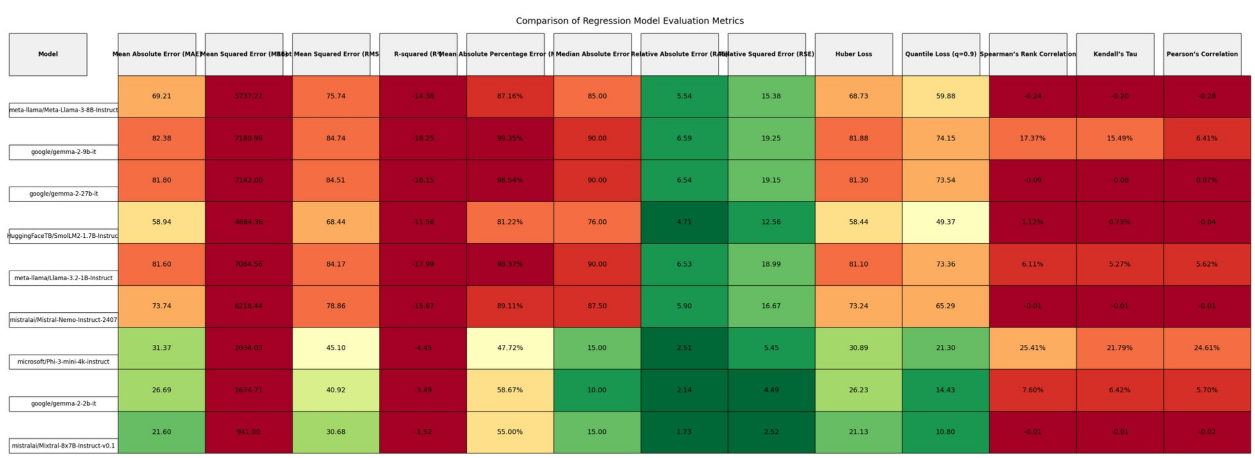
```
59.88100
google/gemma-2-9b-it                        81.8840
74.14560
google/gemma-2-27b-it                       81.3000
73.54000
HuggingFaceTB/SmolLM2-1.7B-Instruct         58.4407
49.37493
meta-llama/Llama-3.2-1B-Instruct            81.1000
73.36000
mistralai/Mistral-Nemo-Instruct-2407        73.2400
65.29400
microsoft/Phi-3-mini-4k-instruct            30.8850
21.29700
google/gemma-2-2b-it                        26.2250
14.42900
mistralai/Mixtral-8x7B-Instruct-v0.1        21.1300
10.80000

                                      Spearman’s Rank Correlation  \
meta-llama/Meta-Llama-3-8B-Instruct                     -0.244107
google/gemma-2-9b-it                                     0.173731
google/gemma-2-27b-it                                   -0.088373
HuggingFaceTB/SmolLM2-1.7B-Instruct                      0.011208
meta-llama/Llama-3.2-1B-Instruct                         0.061110
mistralai/Mistral-Nemo-Instruct-2407                    -0.011392
microsoft/Phi-3-mini-4k-instruct                         0.254131
google/gemma-2-2b-it                                     0.076014
mistralai/Mixtral-8x7B-Instruct-v0.1                    -0.012997

                                      Kendall’s Tau  Pearson’s
Correlation
meta-llama/Meta-Llama-3-8B-Instruct       -0.195584              -
0.283593
google/gemma-2-9b-it                       0.154880
0.064106
google/gemma-2-27b-it                     -0.078579
0.008727
HuggingFaceTB/SmolLM2-1.7B-Instruct        0.007261              -
0.040209
meta-llama/Llama-3.2-1B-Instruct           0.052712
0.056248
mistralai/Mistral-Nemo-Instruct-2407      -0.010028              -
0.008900
microsoft/Phi-3-mini-4k-instruct           0.217891
0.246132
google/gemma-2-2b-it                       0.064222
0.056992
mistralai/Mixtral-8x7B-Instruct-v0.1      -0.010948              -
0.018749
```




```
<ipython-input-27-ac707493c278>:281: UserWarning: Tight layout not
applied. The left and right margins cannot be made large enough to
accommodate all Axes decorations.
  plt.tight_layout()
```



Comparison of Regression Model Evaluation Metrics

Model	Mean Absolute Error (MAE	Mean Squared Error (MSE	Root Mean Squared Error (RMS	R-squared (R²	Mean Absolute Percentage Error (M	Median Absolute Error	Relative Absolute Error (RAE	Relative Squared Error (RSE)	Huber Loss	Quantile Loss (q=0.9)	Spearman’s Rank Correlation	Kendall’s Tau	Pearson’s Correlation
meta-llama/Meta-Llama-3-8B-Instruct	69.21	5737.27	75.74	-14.38	87.16%	85.00	5.54	15.38	68.73	59.88	-0.24	-0.20	-0.28
google/gemma-2-9b-it	82.38	7180.99	84.74	-18.25	99.35%	90.00	6.59	19.25	81.88	74.15	17.37%	15.49%	6.41%
google/gemma-2-27b-it	81.80	7142.00	84.51	-18.15	98.54%	90.00	6.54	19.15	81.30	73.54	-0.09	-0.08	0.87%
HuggingFaceTB/SmolLM2-1.7B-Instruc	58.94	4684.36	68.44	-11.56	81.22%	76.00	4.71	12.56	58.44	49.37	1.12%	0.73%	-0.04
meta-llama/Llama-3.2-1B-Instruct	81.60	7084.56	84.17	-17.99	98.37%	90.00	6.53	18.99	81.10	73.36	6.11%	5.27%	5.62%
mistralai/Mistral-Nemo-Instruct-2407	73.74	6218.44	78.86	-15.67	89.11%	87.50	5.90	16.67	73.24	65.29	-0.01	-0.01	-0.01
microsoft/Phi-3-mini-4k-instruct	31.37	2034.03	45.10	-4.45	47.72%	15.00	2.51	5.45	30.89	21.30	25.41%	21.79%	24.61%
google/gemma-2-2b-it	26.69	1674.71	40.92	-3.49	58.67%	10.00	2.14	4.49	26.23	14.43	7.60%	6.42%	5.70%
mistralai/Mixtral-8x7B-Instruct-v0.1	21.60	941.00	30.68	-1.52	55.00%	15.00	1.73	2.52	21.13	10.80	-0.01	-0.01	-0.02

```
Evaluating meta-llama/Meta-Llama-3-8B-Instruct:   1%|          | 1/100
[00:00<00:21,  4.68it/s]

-----------------------------------------------------------------------
-----
HTTPError                                 Traceback (most recent call
```

```
last)
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_http.py
in hf_raise_for_status(response, endpoint_name)
    405     try:
--> 406         response.raise_for_status()
    407     except HTTPError as e:

/usr/local/lib/python3.11/dist-packages/requests/models.py in
raise_for_status(self)
   1023         if http_error_msg:
-> 1024             raise HTTPError(http_error_msg, response=self)
   1025

HTTPError: 402 Client Error: Payment Required for url: https://api-
inference.huggingface.co/models/meta-llama/Meta-Llama-3-8B-Instruct/
v1/chat/completions

The above exception was the direct cause of the following exception:

HfHubHTTPError                            Traceback (most recent call
last)
<ipython-input-27-ac707493c278> in <cell line: 0>()
    288 for model, metrics in results.items():
    289     # Get the true and predicted values from the evaluation
function
--> 290     tr_scores, pr_scores = evaluate_model(model)  # Get the
true and predicted scores
    291
    292     # Generate indices for x-axis

<ipython-input-27-ac707493c278> in evaluate_model(model_checkpoint)
     57         tr_scores.append(tr_score)
     58
---> 59         pr_score = score_code(pr_code, st_code,
model_checkpoint)
     60
     61         if pr_score is not None:

<ipython-input-26-4c1fc0214ea3> in score_code(prof_code, stud_code,
model)
     38   import re
     39
---> 40   completion = client.chat.completions.create(
     41     model=model,
     42     messages=messages,

/usr/local/lib/python3.11/dist-packages/huggingface_hub/inference/
_client.py in chat_completion(self, messages, model, stream,
frequency_penalty, logit_bias, logprobs, max_tokens, n,
presence_penalty, response_format, seed, stop, stream_options,
```

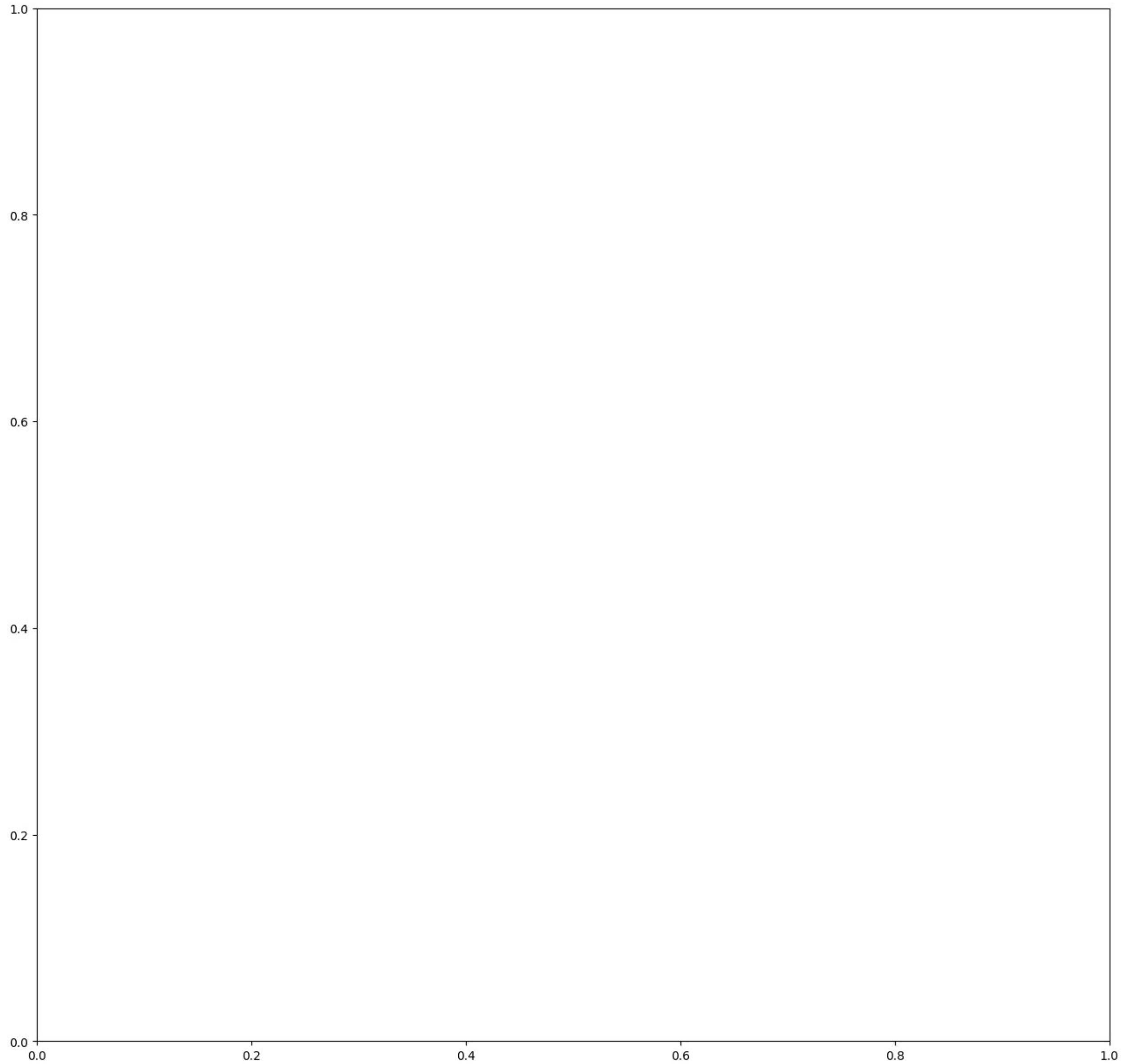
```
temperature, tool_choice, tool_prompt, tools, top_logprobs, top_p)
    968             api_key=self.token,
    969         )
--> 970         data = self._inner_post(request_parameters,
stream=stream)
    971
    972         if stream:

/usr/local/lib/python3.11/dist-packages/huggingface_hub/inference/
_client.py in _inner_post(self, request_parameters, stream)
    325
    326             try:
--> 327                 hf_raise_for_status(response)
    328                 return response.iter_lines() if stream else
response.content
    329             except HTTPError as error:

/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_http.py
in hf_raise_for_status(response, endpoint_name)
    475         # Convert `HTTPError` into a `HfHubHTTPError` to
display request information
    476         # as well (request id and/or server error message)
--> 477         raise _format(HfHubHTTPError, str(e), response) from e
    478
    479

HfHubHTTPError: 402 Client Error: Payment Required for url:
https://api-inference.huggingface.co/models/meta-llama/Meta-Llama-3-
8B-Instruct/v1/chat/completions (Request ID: Root=1-67bf921c-
2b05264342f7dc0e58bae947;2525e272-b9dd-425c-b2d1-3dfc6cc923d4)

You have exceeded your monthly included credits for Inference
Providers. Subscribe to PRO to get 20x more monthly allowance.
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
```

##"

```
```