Task One:

With 20,000 rows, the dataset is relatively large, which allows for more complex models to be trained. Models with more parameters, such as deep neural networks, can potentially capture intricate patterns in the data.

Observations:

With 20,000 rows, the dataset is relatively large, which allows for more complex models to be trained. Models with more parameters, such as deep neural networks, can potentially capture intricate patterns in the data. The dataset has 150 columns, indicating a relatively high-dimensional feature space. The nature of the bacterial properties is not specified. Are they continuous, or categorical, or a mix? No access to a GPU cluster, hence the requirement of a model that can be trained on CPU in a reasonable timeframe. Without a specific target variable mentioned, it is challenging to determine whether it requires a regression or classification approach.

Model: Random Forest Model

Random Forest typically provides good accuracy in various machine learning tasks, including classification and regression. It combines multiple decision trees, leveraging the diversity and collective wisdom of the ensemble to make accurate predictions.

Ensemble of uncorrelated trees: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. The key idea behind Random Forest is to create an ensemble of uncorrelated trees. Each tree is trained on a random subset of the data (bootstrap samples) and uses a random subset of the features. By introducing randomness in both the data and feature selection, each tree in the ensemble learns different patterns and makes diverse predictions.

Capturing diverse patterns: Bacterial properties can be influenced by various factors, and different properties may have different relationships with the features. By training multiple trees on different subsets of the data and features, the Random Forest can capture a wide range of patterns and relationships. This diversity allows the model to capture both general trends and specific dependencies, leading to improved predictive performance.

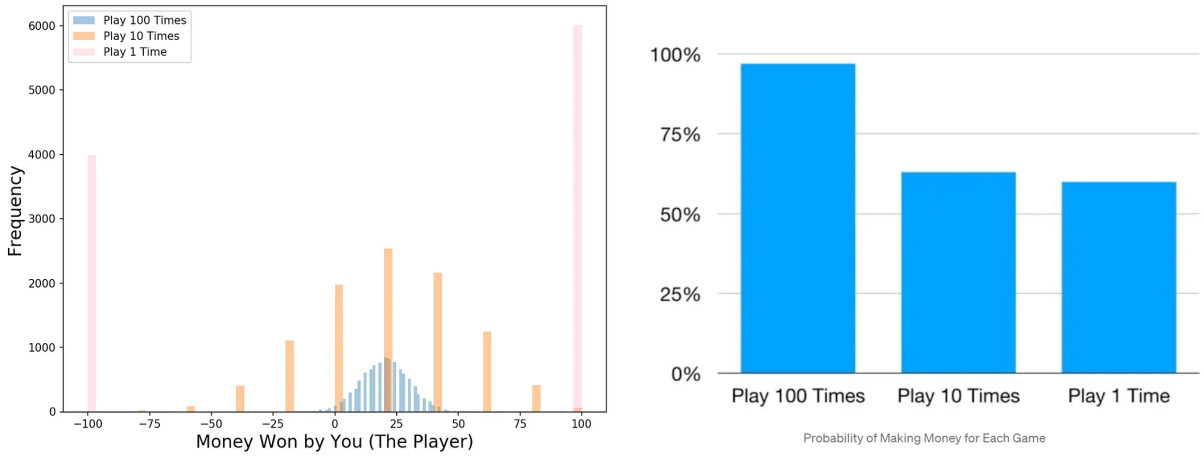
Reducing overfitting: Overfitting occurs when a model learns the training data too well and fails to generalize to unseen data. Random Forest can mitigate overfitting by introducing randomness during training. By using random

subsets of data and features, the model avoids relying too heavily on any single subset or feature, reducing the likelihood of overfitting and improving generalization to unseen data.

A really good simulation is a game where a uniformly distributed random number generator produces a number. If the number generated is greater than or equal to 40, we win (so we have a 60% chance of victory). If it is below 40, we lose. We have three choices:

1. Game 1 — play 100 times, betting $1 each time.
2. Game 2— play 10 times, betting $10 each time.
3. Game 3— play one time, betting $100.

The expected value in all three games is the same, $60. But the outcome distributions are vastly different:



Random forest is the same — each tree is like one play in our game earlier. With a random forest model, our chances of making correct predictions increase with the number of uncorrelated trees in our model.

In the specific task of predicting bacterial properties, incorporating a larger number of uncorrelated trees in the Random Forest ensemble increases the diversity and collective wisdom of the model. This can help capture a broader range of patterns and relationships in the data, leading to improved predictive **accuracy**. Additionally, the ensemble nature of Random Forest allows for better robustness against noise and variability in the dataset.

It's worth noting that there is a **tradeoff** between model complexity and training time. As the number of trees increases, training time may also increase, but this tradeoff is often manageable with Random Forest. By finding an optimal balance between the number of trees and available computational resources, the Random Forest model can

effectively leverage the power of ensemble learning to improve prediction accuracy in the task of predicting bacterial properties.

References:

1 RANDOM FORESTS Leo Breiman Statistics Department University of California Berkeley, CA 94720 January 2001 Abstract Random fore

Understanding Random Forest. How the Algorithm Works and Why it Is… | by Tony Yiu | Towards Data Science

StatQuest: Random Forests Part 2: Missing data and clustering

(incomplete) Task Two:

In a picturesque coastal town, a beachside business owner sought to understand the factors influencing the number of people visiting the beach. With a dataset of normalized weather features, including temperature, precipitation, humidity, wind speed, cloud cover, visibility, snowfall, and UV index, the goal was to predict the crowd size accurately. By analyzing these weather variables and their impact on beach attendance, the business owner aimed to optimize resource allocation, such as staff and amenities, based on the expected number of beachgoers. This would enable them to enhance the overall beach experience and maximize customer satisfaction.

Observations:

Predicting the number of people on the beach is a regression problem because the target variable is a continuous numerical value. Regression models are designed to predict continuous values, whereas classification models are used for tasks with categorical or discrete target variables. The presence of a specific target variable in the second task influences the model selection process by indicating that it is a supervised learning regression problem.

The dataset consists of normalized weather features, including temperature, precipitation, humidity, wind speed, cloud cover, visibility, snowfall, and UV index. These features are continuous and numerical.

Model: Random Forest Model OR Gradient Boosting

Both Gradient Boosting and Random Forest models can be effective for predicting the number of people on the beach.

Task Three:

MS-COCO 2014 dataset — create an AI-based search engine using this dataset. If the input is given as an image, then return text (that most suitably fits that image) and vice versa.

Observations:

The MS-COCO dataset contains a large collection of images along with corresponding textual descriptions. Handling such a vast amount of data requires careful consideration of computational resources, storage, and efficient retrieval techniques. Balancing search accuracy and retrieval speed is vital.

Model:

One of the major limitations of traditional artificial neural networks (ANNs) is their struggle with the computational complexity required for processing image data. While ANNs can effectively handle small image dimensions, such as the 28x28 images in the MNIST handwritten digits dataset, they face challenges when dealing with larger and more complex images.

For instance, if we consider a larger colored image input of 64x64, a single neuron in the first hidden layer would need to accommodate 12,288 weights (64x64x3 for the three color channels). This significantly increases the computational burden on the network. Additionally, to handle such a scale of input, the network itself would need to be much larger than what is used for classifying smaller and grayscale MNIST digits. These factors highlight the drawbacks of using traditional ANN models for processing larger and more complex images.

The computational complexity and increased number of parameters required to process larger and colored images pose challenges for traditional ANN models. Specialized architectures and techniques, such as convolutional neural networks (CNNs) and deep learning, have been developed to address these limitations and achieve better performance on image-related tasks.

Commonly used architectures for image encoders include convolutional neural networks (CNNs), such as VGG16, ResNet, or EfficientNet. For text encoders, popular choices include pre-trained models like BERT, GPT, or Transformer-based architectures.

Image to Text Search:

An input image is processed by an image encoder, which is typically a pre-trained convolutional neural network (CNN) model. The image encoder extracts meaningful visual features from the image, capturing high-level representations of objects, shapes, and textures present in the image. The output of the image encoder is a fixed-length vector or embedding that represents the image's content in a dense feature space.

Once the image is encoded, the dataset in the text-image search engine task becomes similar to other extensive datasets, such as the ones in the first two tasks, in terms of its structure and organization. The image and textual embeddings need to be compared to capture the visual-textual relationship and find meaningful associations.

Text to Image Search:

The text encoder generates a fixed-length representation or embedding for the textual query, capturing the semantic meaning and context of the query. The similarity between the query's embedding and the image embeddings is calculated, and the top-k most similar images are retrieved as search results.

Methodology:

a. Dataset Preparation:

Preprocess the MS-COCO 2014 dataset, which consists of images and textual descriptions, by normalizing and cleaning the data.

b. Image and Text Encoding:

Use pre-trained image encoders, such as convolutional neural networks (CNNs), to extract visual features from the images in the dataset. Apply text encoders, such as language models or word embedding models, to convert the textual descriptions into fixed-length text embeddings.

c. Structured Meaning Module:

Incorporate a structured meaning module that combines visual and semantic information. This module can utilize external knowledge sources, such as semantic graphs or structured ontologies, to enhance the representation and alignment of visual and textual data.

d. Learning Unified Embeddings:

Train the model to learn unified visual-semantic embeddings by optimizing a suitable objective function, such as a combination of cross-modal retrieval loss and alignment loss. The objective function encourages the embeddings to capture the semantic relationships between visual and textual elements.

e. Cross-Modal Retrieval:

For image-to-text search, encode an input image using the image encoder and find the most suitable textual descriptions by measuring the similarity or distance between the image representation and the text embeddings. For text-to-image search, encode a textual query using the text encoder and retrieve the most relevant images from the dataset by comparing the query representation with the encoded image representations.

f. Evaluation:

Evaluate the performance of the search engine using standard evaluation metrics like precision, recall, mean average precision (MAP), or normalized discounted cumulative gain (NDCG). Conduct experiments on relevant benchmark datasets, such as MS-COCO, to assess the quality and relevance of the search results.

References:

Unified Visual-Semantic Embeddings: Bridging Vision and Language With Structured Meaning Representations

CS101 - Image encoding

(PDF) An Introduction to Convolutional Neural Networks

[1405.0312] Microsoft COCO: Common Objects in Context

But what is a neural network? | Chapter 1, Deep learning

Task Four:

You have a dataset consisting of three columns — two random 3 × 3 matrices, and their matrix product. Assume you have sufficient data to train a model (in case you don't you can always synthetically generate more). You are now given a test set of matrices in a similar format. What technique will you use — if you want to optimize for speed? What if you want to optimize for accuracy? What if you do or do not have

access to a GPU?

Model:

To optimize for speed:

Linear Regression: Linear regression is a simple and computationally efficient model that can capture linear relationships between input features and the target variable. In this case, linear regression can learn the linear mapping between the input matrices and their matrix product. It is a straightforward model to implement and train, and it can provide reasonably fast predictions. (Refer to: Operations Research OR Forum—An Algorithmic Approach to Linear Regression)

To optimize for accuracy:

MLP is a type of feedforward neural network with multiple hidden layers. It can capture both linear and non-linear relationships between the input matrices and their matrix product. MLPs have the advantage of being more expressive and flexible in learning complex patterns in the data. However, compared to linear regression, MLPs may require more computational resources and training time.

1. Activation Functions: While the Perceptron typically uses a threshold-based activation function like step function, the MLP can use any differentiable activation function. Common choices include the Rectified Linear Unit (ReLU), sigmoid, or hyperbolic tangent (tanh). These activation functions introduce non-linearity to the model, allowing it to learn complex patterns and relationships in the data.

2. Feedforward Propagation: In the MLP, information flows in a forward direction through the layers, from the input layer to the output layer. Each neuron in a layer receives inputs from the previous layer, computes a weighted sum of the inputs, applies the activation function, and passes the result to the next layer. This process is repeated for all layers until the output layer is reached.

3. Backpropagation: Backpropagation is the key mechanism for training the MLP. It involves computing the gradient of the cost function with respect to the network's weights. The cost function is typically the mean squared error (MSE) between the predicted output and the true output. The gradient is then propagated backward from the output layer to the hidden layers, allowing the weights to be adjusted based on the error

signal. This iterative process of forward propagation and backward propagation is performed until convergence, where the weights have reached a satisfactory solution.

4. Gradient Descent: Backpropagation relies on the principle of gradient descent to update the weights. The gradient of the cost function with respect to the weights is computed using the iterative chain rule. This gradient represents the direction and magnitude of the weight updates needed to minimize the cost function. Gradient descent is then used to adjust the weights by taking steps proportional to the negative gradient, gradually moving towards the optimal weights that minimize the cost.

5. Convergence and Learning: The backpropagation process continues iteratively until a convergence criterion is met. This criterion is often based on the change in the cost function or the gradient magnitude falling below a specified threshold. As the MLP iteratively updates its weights, it learns to better approximate the desired output and minimize the cost function. Through this learning process, the MLP can capture complex relationships and make accurate predictions on unseen data.

6. Trade-off between Speed and Accuracy: The choice of optimizing for speed or accuracy depends on the specific requirements of the task. When optimizing for speed, simpler MLP architectures with fewer hidden layers and neurons can be used. These models are computationally less demanding and faster to train but may sacrifice some accuracy. On the other hand, optimizing for accuracy may involve larger and deeper MLP architectures, which can capture more complex patterns but require more computational resources and training time.

In summary, the Multilayer Perceptron (MLP) with the backpropagation algorithm is a powerful neural network architecture for learning complex relationships in data. Through forward propagation and backpropagation, the MLP can process inputs, compute weighted sums, apply activation functions, and adjust weights to minimize the cost function. The choice of activation function, convergence criteria, and model complexity influences the speed and accuracy trade-off in MLP training. Overall, the MLP with backpropagation enables the network to learn complex patterns and optimize its weights to minimize the cost function.

Considering the dataset size and the nature of the task, linear regression would be a simpler and more efficient choice. It can provide reasonably accurate predictions while being computationally efficient. However, if there is a

need for capturing more complex relationships in the data and achieving higher accuracy, MLP can be a viable option, although it may require more computational resources and training time. It's important to note that the final choice depends on the specific requirements and constraints of the task, such as the desired level of accuracy, available computational resources, and time constraints.

References:

Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis | by Carolina Bento | Towards Data Science

Operations Research OR Forum—An Algorithmic Approach to Linear Regression