¡Por supuesto! Aquí hay una estructura básica que puedes seguir para tu presentación sobre Machine Learning:

1. Introducción

- Breve explicación del problema que se va a resolver

- Objetivos de la presentación

2. Recopilación y análisis de datos

- Fuente de datos

- Proceso de limpieza y transformación de datos

- Visualización de datos

3. Selección de algoritmos y entrenamiento del modelo

- Selección de algoritmos de Machine Learning adecuados para el problema

- Preprocesamiento de datos y división en conjunto de entrenamiento y prueba

- Entrenamiento del modelo y ajuste de hiperparámetros

4. Evaluación del modelo

- Métricas de evaluación utilizadas

- Interpretación de los resultados obtenidos

5. Conclusiones y próximos pasos

- Resumen de los resultados y conclusiones

- Limitaciones del modelo y posibles mejoras

- Próximos pasos a seguir para mejorar el modelo

6. Referencias bibliográficas

- Lista de fuentes utilizadas durante el proceso de investigación

Recuerda que es importante que la presentación sea clara y concisa, y que esté dirigida a una audiencia no técnica. Asegúrate de incluir ejemplos y visualizaciones para hacer más fácil la comprensión de los conceptos. ¡Buena suerte con tu presentación!

A continuación te presento algunos modelos de machine learning que se incluyen en cada uno de los grupos que mencionaste, así como algunas de las librerías populares que se pueden utilizar para implementarlos en Python:

**Regression Algorithms:** Los algoritmos de regresión se utilizan para predecir valores numéricos continuos. Algunos ejemplos de algoritmos de regresión son:

* Regresión lineal (Linear Regression)
* Regresión logística (Logistic Regression)
* Regresión de Ridge (Ridge Regression)
* Regresión de Lasso (Lasso Regression)
* Regresión de Elastic Net (Elastic Net Regression)

Las librerías populares para la implementación de algoritmos de regresión en Python son:

* scikit-learn (sklearn)
* Statsmodels
* TensorFlow
* PyTorch

**Regularization Algorithms:** Los algoritmos de regularización se utilizan para evitar el sobreajuste y mejorar la capacidad de generalización de los modelos de machine learning. Algunos ejemplos de algoritmos de regularización son:

* Regresión de Ridge (Ridge Regression)
* Regresión de Lasso (Lasso Regression)
* Regresión de Elastic Net (Elastic Net Regression)
* Penalización de normas (Norm Penalties)

Las librerías populares para la implementación de algoritmos de regularización en Python son las mismas que para los algoritmos de regresión.

**Tree Based Models:** Los modelos basados en árboles son algoritmos de machine learning que utilizan árboles de decisión para modelar relaciones no lineales entre las variables. Algunos ejemplos de modelos basados en árboles son:

* Árboles de decisión (Decision Trees)
* Bosques aleatorios (Random Forests)
* Gradient Boosting Machines (GBM)
* eXtreme Gradient Boosting (XGBoost)
* LightGBM
* CatBoost

Las librerías populares para la implementación de modelos basados en árboles en Python son:

* scikit-learn (sklearn)
* TensorFlow
* PyTorch
* XGBoost
* LightGBM
* CatBoost

**Misc Models:** En este grupo se incluyen modelos que no encajan en las categorías anteriores. Algunos ejemplos son:

* Máquinas de vectores de soporte (Support Vector Machines)
* K-vecinos más cercanos (K-Nearest Neighbors)
* Redes Bayesianas (Bayesian Networks)
* Redes neuronales de retropropagación (Backpropagation Neural Networks)

Las librerías populares para la implementación de modelos misceláneos en Python son:

* scikit-learn (sklearn)
* TensorFlow
* PyTorch
* Keras

**Neural Networks and Deep Learning:** Las redes neuronales y el aprendizaje profundo son técnicas de machine learning que utilizan modelos inspirados en la estructura y funcionamiento del cerebro humano. Algunos ejemplos de modelos de redes neuronales son:

* Perceptrón multicapa (Multilayer Perceptron)
* Redes neuronales convolucionales (Convolutional Neural Networks)
* Redes neuronales recurrentes (Recurrent Neural Networks)
* Redes neuronales generativas adversarias (Generative Adversarial Networks)

Las librerías populares para la implementación de redes neuronales y aprendizaje profundo en Python son:

1. TensorFlow: es una librería de código abierto desarrollada por Google que se utiliza para la creación y entrenamiento de modelos de redes neuronales y aprendizaje profundo. También proporciona herramientas para la visualización de modelos y datos.
2. Keras: es una librería de alto nivel para la creación y entrenamiento de modelos de redes neuronales y aprendizaje profundo. Está diseñada para ser fácil de usar, modular y extensible.
3. PyTorch: es una librería de aprendizaje automático de código abierto desarrollada por Facebook que se utiliza para la creación y entrenamiento de modelos de redes neuronales y aprendizaje profundo. También proporciona herramientas para la visualización de modelos y datos.
4. Theano: es una librería de aprendizaje automático de código abierto que se utiliza para la creación y entrenamiento de modelos de redes neuronales y aprendizaje profundo. Fue desarrollada por el equipo de investigación de aprendizaje profundo de la Universidad de Montreal y es muy popular en la comunidad de investigación de aprendizaje profundo.
5. Caffe: es una librería de aprendizaje automático de código abierto que se utiliza principalmente para la creación y entrenamiento de modelos de redes neuronales convolucionales. Fue desarrollada por el equipo de investigación de aprendizaje profundo de la Universidad de California, Berkeley, y se utiliza ampliamente en la industria para aplicaciones de visión por computadora.
6. MXNet: es una librería de aprendizaje automático de código abierto que se utiliza para la creación y entrenamiento de modelos de redes neuronales y aprendizaje profundo. Fue desarrollada por el equipo de investigación de aprendizaje profundo de Amazon y es conocida por su escalabilidad y eficiencia.
7. TensorFlow.js: es una librería de aprendizaje automático de código abierto que se utiliza para la creación y entrenamiento de modelos de redes neuronales y aprendizaje profundo en el navegador web. Permite la implementación de modelos de aprendizaje profundo en la web para aplicaciones de inteligencia artificial.

Estas son solo algunas de las librerías populares para la implementación de redes neuronales y aprendizaje profundo en Python. Hay muchas más disponibles en la comunidad de aprendizaje automático, y la elección depende del problema que estés intentando resolver y de tu nivel de experiencia en el campo.

Bagging (Bootstrap Aggregating) es una técnica de aprendizaje automático que utiliza múltiples modelos para mejorar la precisión y la estabilidad de las predicciones. La idea fundamental detrás de bagging es entrenar múltiples modelos utilizando diferentes muestras de datos y luego combinar las predicciones de los modelos individuales para producir una predicción final.

Una forma en que bagging se puede utilizar como herramienta de optimización de modelos es mediante el uso de la técnica de Bootstrap para generar múltiples muestras de datos. La técnica de Bootstrap consiste en tomar múltiples muestras aleatorias con reemplazo del conjunto de datos original. Cada muestra bootstrap se utiliza para entrenar un modelo individual. Al entrenar los modelos en muestras de datos diferentes, se puede reducir la variabilidad de los modelos y mejorar la precisión de las predicciones.

Además, al combinar los modelos individuales, se puede reducir el riesgo de sobreajuste y aumentar la estabilidad de las predicciones. Los modelos individuales en un conjunto bagging no necesariamente tienen que ser idénticos. De hecho, es común utilizar diferentes algoritmos de aprendizaje automático o diferentes hiperparámetros en cada modelo individual para aumentar la diversidad en el conjunto bagging.

En resumen, Bagging es una técnica útil de optimización de modelos que puede mejorar la precisión y la estabilidad de las predicciones mediante el entrenamiento de múltiples modelos individuales en diferentes muestras de datos y la combinación de las predicciones de los modelos individuales.

En general, el RMSE y el MAE son las métricas que se utilizan para evaluar modelos de regresión y son especialmente útiles cuando los valores extremos son importantes. Ambas métricas miden la diferencia entre las predicciones del modelo y los valores reales en la misma escala que el target, lo que puede ser útil para evaluar el error en los valores extremos.

El RMSE (Root Mean Squared Error) es una medida de la raíz cuadrada de la media de los errores al cuadrado. Esta métrica penaliza los errores grandes más que los pequeños. Por lo tanto, si los valores extremos son importantes para tu problema, deberías considerar el RMSE como una métrica importante a tener en cuenta.

El MAE (Mean Absolute Error) es una medida de la media de los errores absolutos. A diferencia del RMSE, el MAE no penaliza los errores grandes más que los pequeños, por lo que si los valores extremos no son demasiado importantes en tu problema, el MAE puede ser una buena métrica a considerar.

En general, se recomienda utilizar ambas métricas para evaluar el rendimiento de un modelo de regresión, especialmente cuando se trata de problemas en los que los valores extremos son importantes.