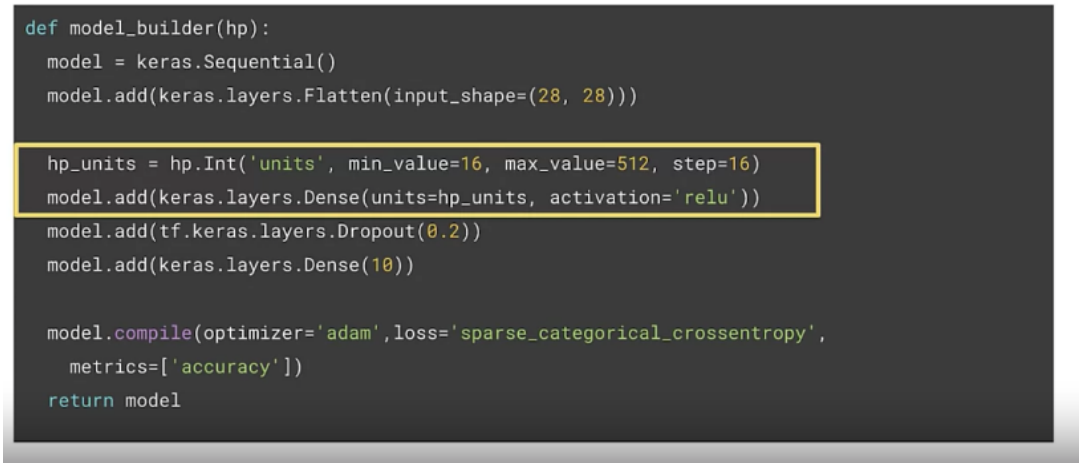
Machine Learning Modeling Pipelines in Production

Week 1 - Neural Architecture Search

This week they talked about Hyperparameter Tuning, AutoML and Search Strategies.

Hyperparameter Tuning

The tool used for hyperparameter tuning in the class was the Keras Tuner (import kerastuner as kt). One example for changing the number of nodes in a certain layer can be seen below.



```
def model_builder(hp):
  model = keras.Sequential()
  model.add(keras.layers.Flatten(input_shape=(28, 28)))

  hp_units = hp.Int('units', min_value=16, max_value=512, step=16)
  model.add(keras.layers.Dense(units=hp_units, activation='relu'))
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(keras.layers.Dense(10))

  model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
  return model
```

Search Strategies

They talked about 5 different strategies, these were:

1. Grid search
    a. Exhaustive search approach, suitable for smaller search spaces
2. Random search (randomize search)
    a. Random search approach, suitable for smaller search spaces
3. Bayesian Optimization
    a. Assumes a specific probability distribution is underlying the performance
4. Evolutionary Algorithms
    a. Select parents -> parents + offspring -> select + remove -> replace with offspring
5. Reinforcement Learning
    a. Agents goal is to maximize a reward, the performance estimation strategy determines the reward

The assignment was about training a model in Google Cloud environment for classifying images of clouds.

Week 2 - Model Resource Management Techniques

This week they talked about Dimensionality Reduction and Quantization and Pruning.

Dimensionality Reduction

Curse of dimensionality states that having more features is bad (redundant features, noise added, hard to interpret and visualize, and hard to store and process the data).

There are a few options to reduce dimensionality:

- Manual dimensionality reduction: tabular (aggregate, combine and decompose), text-extract context indicators and image-prescribe filters for relevant structures
- Principal component analysis (PCA): minimization of the orthogonal distance, it accounts for variance of data in as few dimensions as possible using linear projections

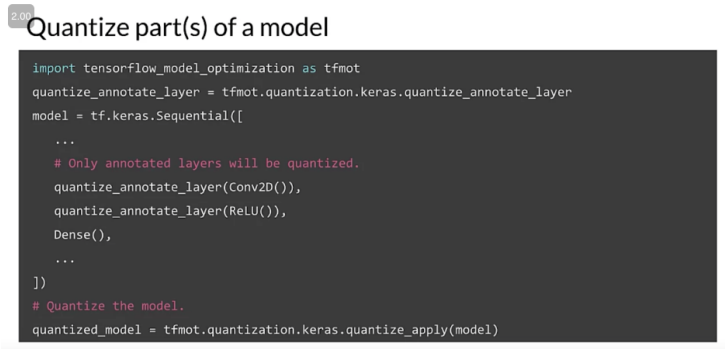
Quantization and Pruning

Quantization is a technique to shrink model file size, reduce computational resources and make models run faster and use less power with low-precision.

It's possible to also do post-training quantization, for example:

- Dynamic range quantization
- Full integer quantization
- Float16 quantization

The side effects could be a reduced precision representation, small loss in model accuracy, and joint optimization for model and latency.



Pruning aims to reduce the number of parameters and operations involved in generating a prediction by removing network connections. Thus, causing better storage and/or transmission, gain speedups in CPU and some ML accelerators, can be used in tandem with quantization to get additional benefits and unlock performance improvements.

Week 3 - High-Performance Modeling

This week they talked about High-Performance Modeling and Knowledge Distillation.

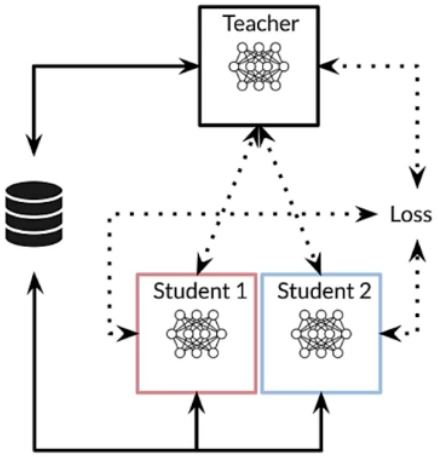
High-Performance Modeling

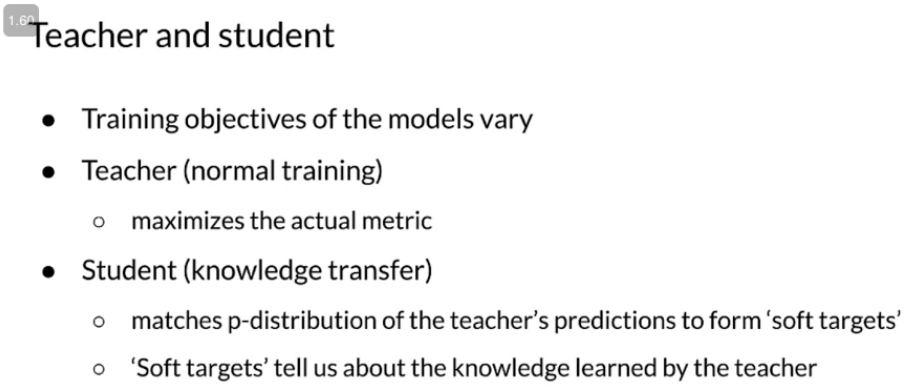
One option is to use distributed training, there are two types:

- Data parallelism: models are replicated into different accelerators and the data is split between them
- Model parallelism: when models are too large to fit on a single device and it can be divided into partitions, assigning different partitions to different accelerators

Knowledge Distillation

Knowledge distillation it's a way to train small models to mimic bigger models. The idea is as follows: create a simple "student" model that learns from a complex "teacher" model.





Week 4 - Model Analysis

This week they talked about Model Analysis Overview, Advanced Model Analysis and Debugging, and Continuous Evaluation and Monitoring.

Model Analysis Overview

What is next after model training/deployment?

- Is the model performing well?
- Is there scope for improvement?
- Can the data change in the future?
- Has the data changed since you created your training dataset?

For black box evaluation there is a library called TensorBoard from TensorFlow that helps make sense of the numbers and loss during iterations.

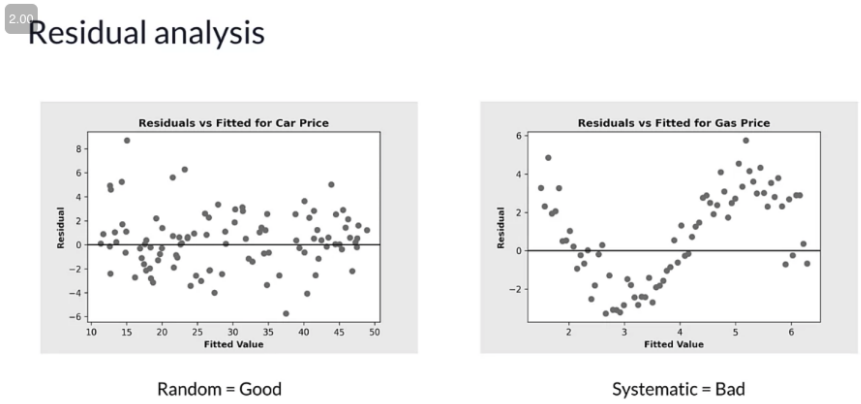
Advanced Model Analysis and Debugging

In this part they suggest using TFMA (TensorFlow Model Analysis), where one could compare how the model performance varies in different slices of data with different metrics too, or also how to track metrics over time.

Debugging helps understanding the model robustness, and robustness is much more than generalization, is understanding if the model is accurate even for slightly corrupted input data.

For Sensitivity analysis one could use the What-If tool from TensorFlow, the idea is to simulate data of your choice and see what your model predicts, see how the model reacts to data which has never been used before.

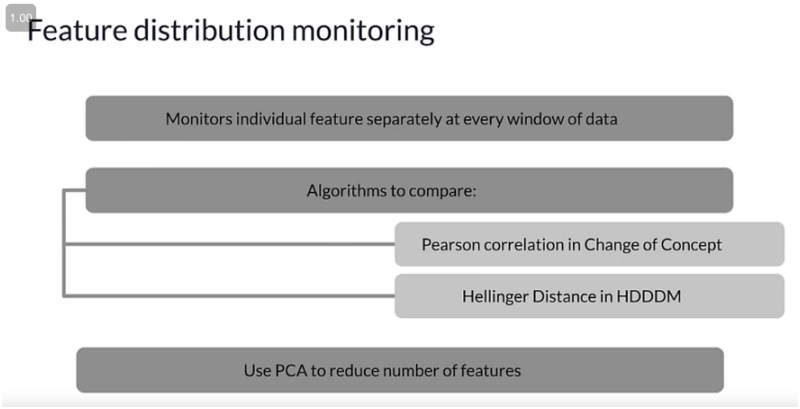
For residual analysis, one should try to guess if for specific inputs, the model is systematically not making proper predictions.



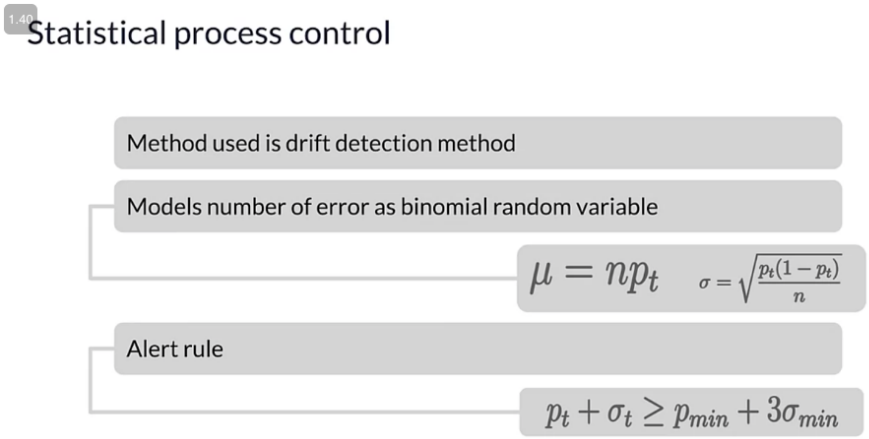
Finally, regarding fairness, it compares model performance across subgroups to other models.

Continuous Evaluation and Monitoring

There are a few things to monitor features, such as:



He briefly talked about Statistical process control, it was unclear if it should be used for the models output or for the variables input.



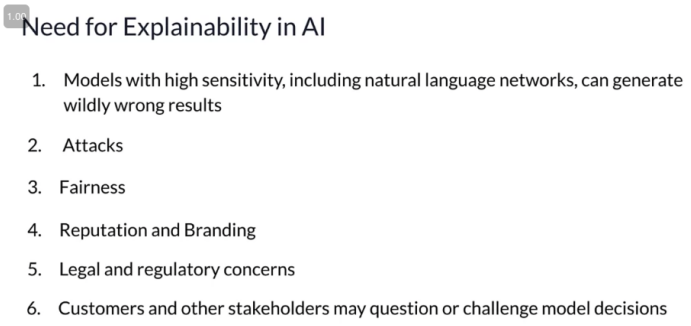
How often should you retrain your models?

- It depends on the rate of change
- If possible, automate the management of detecting model drift and triggering alerts when the model needs to be retrained

Week 5 - Interpretability

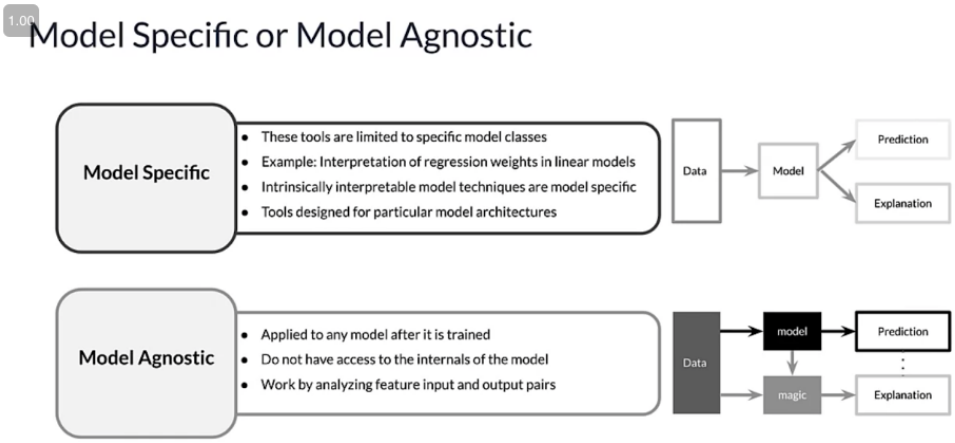
This week they talked about Explainable AI, Interpretability, and Understanding Model Predictions.

Explainable AI



Interpretability

There are two types of model interpretation methods: Model Specific and Model Agnostic.



Also, there are local or global explanations. Local is when the interpretation method explains an individual prediction. Global is when the interpretation method explains the entire model behaviour.

Understanding Model Predictions

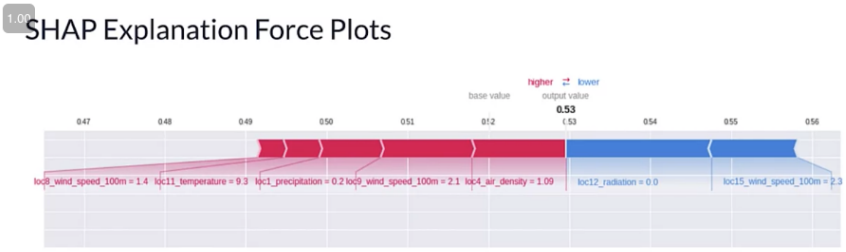
There are a few methods to help understanding the model predictions, such as:

- **Partial dependence plots:** shows the marginal effect of one or two features have on the model result, also whether the relationship between the targets and the feature is linear, monotonic or more complex
- **Permutation feature importance:** estimate the original model error for each feature changing the value (input)
- **Shapley value:** assigns payouts to players (variables) depending on their contribution to the total payout (prediction)
- **LIME:** implements local surrogate models, interpretable models that are used to explain individual predictions

For the Shapley values there is the SHAP (SHapley Additive exPlanations) library which includes:

- TreeExplainer: high-speed exact algorithm for tree ensembles
- DeepExplainer: high-speed approximation algorithm for SHAP values in deep learning models
- GradientExplainer: combines ideas from Integrated Gradients, SHAP, and SmoothGrad into a single expected value equation
- KernelExplainer: uses a specially-weighted local linear regression to estimate SHAP values for any model

From the SHAP library, one can extract the local explanations:



And also the global explanations: