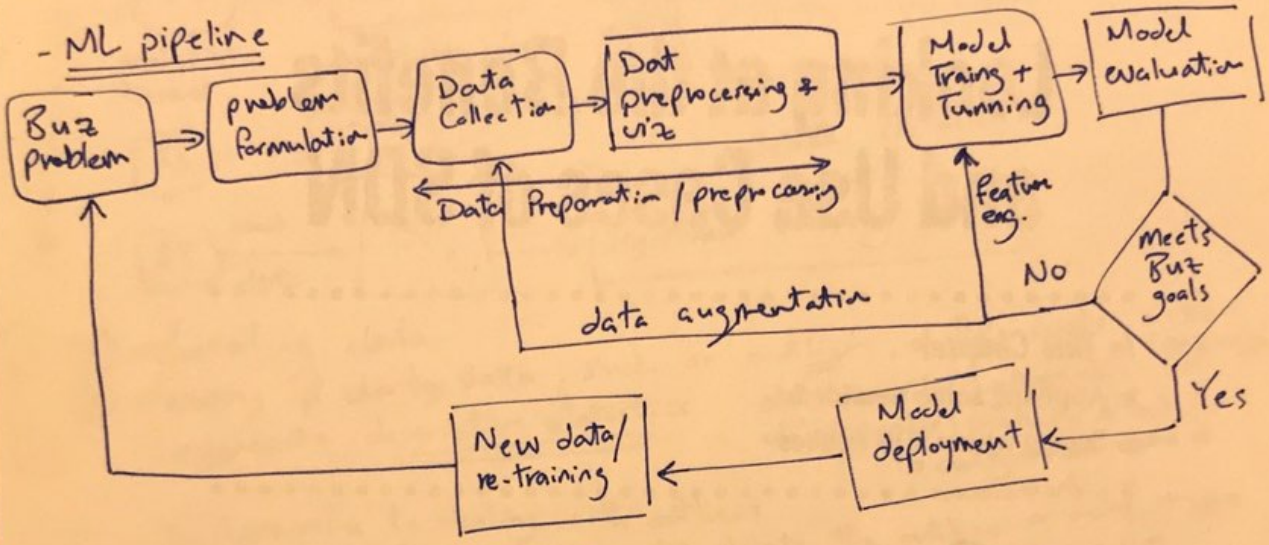
* ML pipeline Notes

- Supervised learning → Binary classification / multiclass / Regression
- Unsupervised learning → to uncover & create the labels itself
  - ↳ clustering / association
  - ↳ core use case is anamoly detection (outliers)
- reinforcement learning → learn through trial / error
  - → Agent / environment / action / reward or penalty

- ML pipeline



- instance types
  1. t family → short jobs/notebooks
  2. m " → standard cpu to memory ratio
  3. r " → memory optimized
  4. c " → compute "
  5. p " → accelerated computing (training/inference)
  6. g " → accelerated inference

  } ML Training DL+

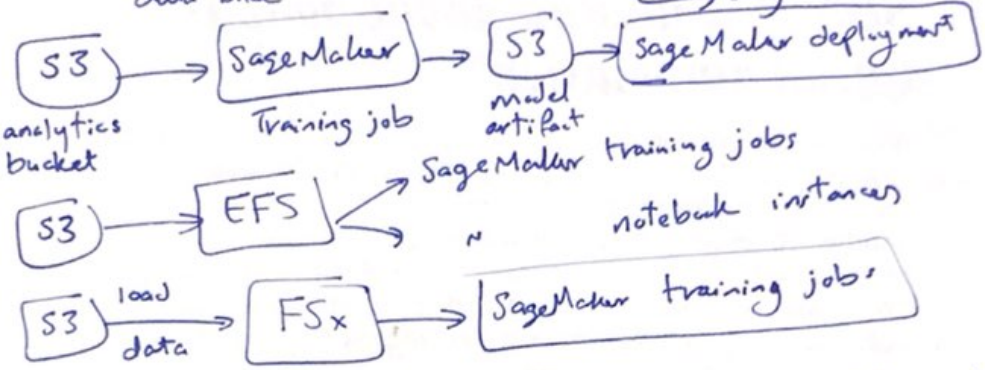
  7. Elastic Inference → inference / cost effective
- Amazon SageMaker Ground Truth → to label the data by human

- Problem formulation
  1. defining the buz problem
  2. what's the buz goal / outcome you need to reach
  3. what does success look like / How you define success?
  4. Now, it's time to decide which ML model to use / (ML problem) what model do we choose?
     Tip: Avoid overcomplicating the problem, and also avoid loring information
  5. Get understand of your data → How much data do you have?
     → Where your data is?
     ↳ Do you have access to that data?
  6. To get what you need from the data → Get a domain expert, with domain knowledge, you can begin to determin the features and target data.

⑦ evaluate the quality of your data and start identifying features and labels you already have

- Data preprocessing

① Store in data lake
↳ use S3 with data lake
↳ single source of truth



② reformating data
③ cleaning up dirty data, such as outliers, diff. scale, diff. language
↳ generate descriptive statistics
df.describe() → num data
df.describe(include='all')
↳ categorical data + numerical "

↳ Approaches to dealing with outliers
⇒ Artifical outlier ⟹ delete the outlier or using mean
⇒ Natural " → Transform the outlier ⇒ taking a (log)
↳ impute a new value by using mean

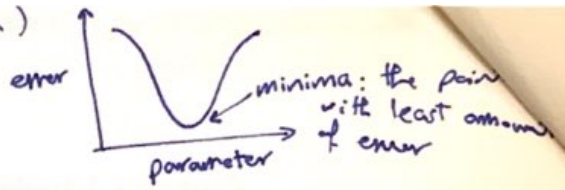
↳ imputing missing numerical data
↳ visualize categorical data ⇒ df.plot.bar()
" numerical " ⇒ histogram, density plot, box plot
↳ Be aware of highly correlated features,
↳ Correlation matrix ↳ leads to poor model performance

- Model Training

→ Amazon SageMaker built-in algorithms support → CSV, Record IO protobuf
→ " " " " → label on the left
→ Validation techniques
① Simple hold-out
② K-fold cross-validation
- average the same model results
- compare results between diff. models
③ Stratified k-fold cross-validation
④ iterated k-fold validation with shuffling

k↑ = time↑ error variance in test data ↑ bias ↓

→ loss function (objective function)
↳ metric used to update the weights after each iteration



↳ RMSE
↳ log likelihood loss ⇒ cross-entropy

→ How to find the minim in a more efficient way?
↳ Gradient descent ⇒ the simplest optimization technique
↳ high learning rate ⇒ overshoot the minima
↳ low " " ⇒ model will take time to reach to the minima

step size ←

Gradient Descent Drawbacks:
update happen very slowly ← ① update parameters only after a pass thru all the data (epoch)
② can't be used for large data to fit into memory entirely
③ can stuck at local minima

All of these drawbacks can be mitigated by "Stockastic Gradient Descent" or SGD
↳ one drawback in SGD, it'll oscillate in diff directions

→ Another technique is to use something in between, "mini-batch gradient descent"

→ Parameters & hyperparameters → values are set before training the model & tuned by user before training the model. external to a model.
↓
learned and updated during training & tuned automatically by the model

→ create training job in SageMaker
↳ fit() method to do the actual training. The fit() API calls the amazon SageMaker CreateTrainingJob API to start model training.

- Model Evaluation
→ validation set ⇒ to improve the model
test set ⇒ to evaluate the predictive quality of the trained model
→ underfitting ⇒ decrease the amount of regularization used
overfitting ⇒ increase " " " "

→ Variance: How dispersed your predicted values are.
Bias: gap between predicted value and actual value

→ Which metrics are most appropriate?

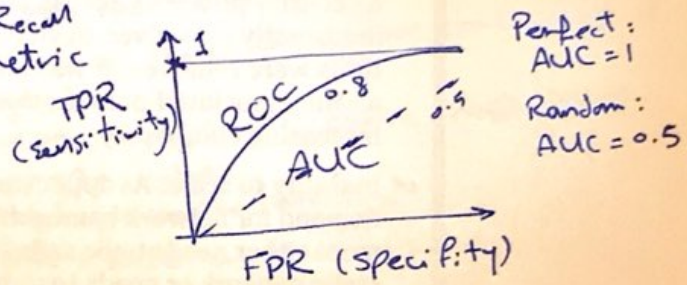
classification problem	Regression problem
- Accuracy - Precision - Recall - F1 score - AUC/ROC	MSE R squared

→ Recall (sensitivity) = $\frac{TP}{TP+FN}$ +ve actual (illness or not)

Precision = $\frac{TP}{TP+FP}$ +ve predicted

F1 score = $2\frac{Precision \cdot Recall}{Precision + Recall}$

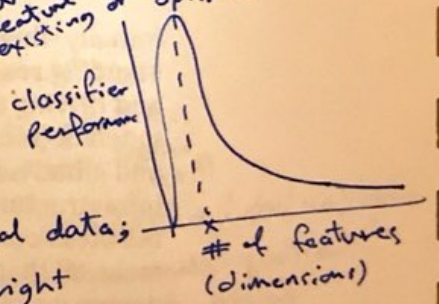
→ AUC-ROC ⇒ evaluation metric



- Feature Engineering

→ Feature engineering components:
① feature extraction } Reduce dimensionality of your dataset
② " selection
③ " creation/transformation → creating/generating new feature from existing ones

→ Curse of dimensionality
→ with structure data, techniques like PCA, and/or T-SNE to reduce the dimensionality



→ Techniques of handling features with numerical data;
① logarithmic transformation ⇒ reduce the right skewness of features
② Square root / cube root
⇒ handling high variance
③ Binning ⇒ convert continuous variable into a categorical variable
④ Scaling → Mean/Variance standardization
→ MinMax Scaling
→ Maxabs Scaling
→ Robust scaling
→ Normalizer

→ Categorical data may need to be converted to numerical data
  ↳ Ordinal Small / medium / large
  ↳ Nominal ⇒ one-hot encoding

-Hyperparameter tunning

→ Hyperparameter categories ⇒ Model ⇒ # of layers & activation function
  ⇒ Optimizer ⇒ ex; SGD, ADAM, HE
  ⇒ Data ⇒ data augmentation (cropping, resize, ...

→ Methods for tuning hyperparameters
  ↳ manually
  ↳ Grid Search (trained and scored for each possible set of hyper.)
    ↳ can be very inefficient
  ↳ Random Search (trained and scored on random combinations of hyper.)
  ↳ Bayesian Search ⇒ use regression to choose next set of hyper. values

→ SageMaker offers automated hyperparameter tunning.
  ↳ for. ex. objective metric : AUC or accuracy ...

-Model Deployment

→ Amazon SageMaker provides:
  * deploy with one click or a single API call
  * Auto scaling
  * Model hosting services
  * https endpoints that can host multiple models

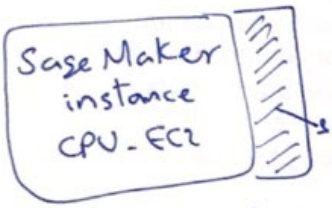
→ Production variants ⇒ Blue / Green deployment
  Blue ↓ live production env. version (n)
  Green ↓ an exact copy of this env. that runs version (n+1)
  ⇒ Canary deployment ⇒ (50/50) → (60/40) → ...
    ↳ route a small portion of users to green
    ↳ the same as A/B testing

→ Use SageMaker batch transform to get predictions for an entire dataset, or you can setup a persistant endpoint to get one prediction at a time using sageMaker hosting services

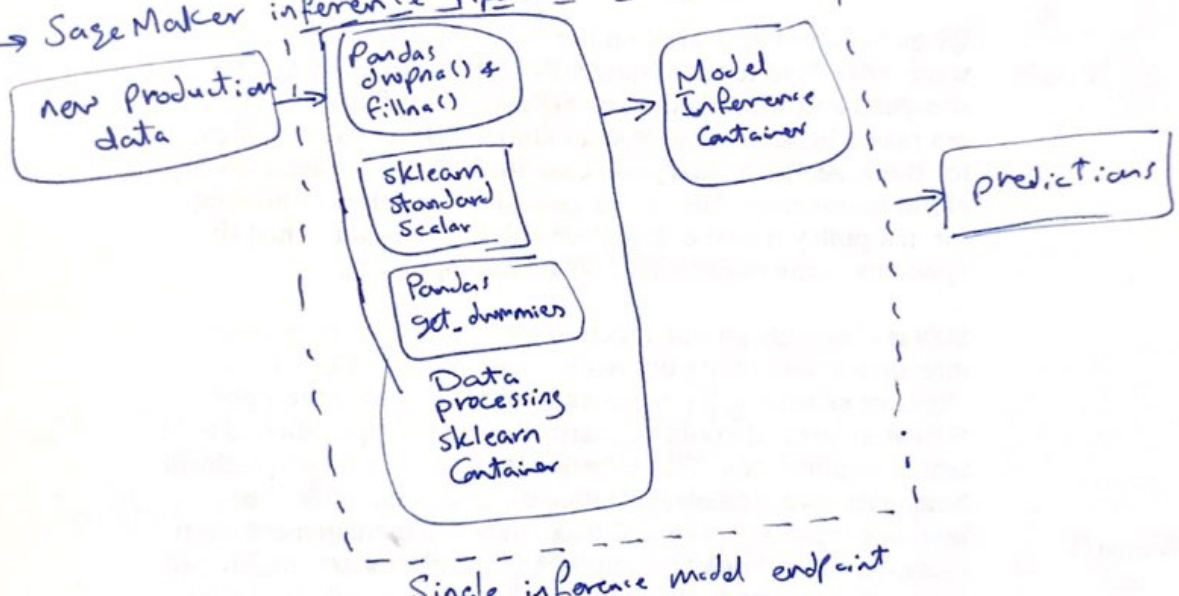
→ Use Batch Transform when you:
- Want to get inferences for an entire dataset and index them to serve inferences in real time
- don't need a persistent endpoint
- " " sub-second latency
- Preprocess your data before using it to train a new model

→ - Batch transform is for serving predictions on a scheduled basis
- Endpoints are best for online inferencing when data is coming from the internet and you need to service predictions in real-time

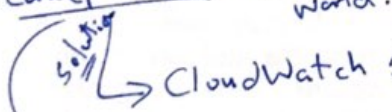
→ Consider AWS Auto scaling only in inference
↳ dynamic / unpredictable capacity needs

→ 

→ Sage Maker inference Pipelines



→ Monitoring your model
- concept drift ⇒ due to gradual misalignment of the model and real world. ML model declines over time in production



→ How often should you retrain your model?
- static env. ⇓ use cron job / cloudwatch events daily / weekly / ...
- Dynamic env. ⇓ monitor → notified → time to retrain

→ Sage Maker continuously monitoring models in production