1. Explore and analyze data in Python.

→ numpy array more suited for numerical analysis than Python lists.
→ shape() → (n, m) (rows, columns)
→ pandas dataframes for tabular data.
pd.Dataframe({'_': ___}) → dict.
→ Difference between loc and iloc.
loc = find rows based on index 'label'
loc[0:5] returns 6 rows → index 0, 1, 2, 3, 4, 5
iloc = find rows based on the positions
iloc[0:5] returns 5 rows → 0, 1, 2, 3, 4
→ handling missing values. fillna(), dropna()
axis=0 = rows
axis=1 = columns.
→ aggregations in pandas
→ Matplotlib figures

Statistical Analyses —

- Plot histograms, boxplots, measure central tendency.
- plot p.d.f.
- find outliers.
- Range, std.dev, variance
- scipy package
- 68%, 95%, 99% rule of normal distribution,
- scale data using sklearn's min-max scaler for comparing values on different scale!
- corr() function - "Correlation ≠ causation".
- fit regression line $y = mx + b$, stats package (in regress() function.
- "residual" or error.

2. Train and evaluate regression models.

(Theory)

- Train, test split.
- MSE, RMSE (on the same scale of target)
- R^2 squared (coefficient of determination)
- what does MSE, RMSE signify? It is not accuracy, how to read it?
- Check for residual plots is very important.
- R^2 = measures the amount of variance that can be explained by the model.
- what are the assumptions of a regression model?
- Statsmodel API better than sklearn
- Explain results of statsmodel API. t-test? p-values?

(Jupyter)

- Bike rental data. predict the continous label "number of rentals".
- Data exploration? what is the goal?
  - understand relationship between it attributes?

- apparent correlation between features!
- fix data issues (missing, errors, outliers)
- derive new features i.e feature engineering.
- Normalize numeric features.
- Encode categorical features.

- date → day of month new feature.
- .describe() on the dataframe.
- look at the distribution of "target" for outliers.
- perform the same for numeric features.
- plot value_counts() for categorical features.
- look at the relationship with target
  - numeric features - plot correlation
  - categorical features - plot group by boxplots
  - look for patterns or trends.

- fit the model. look at scatter of predicted and actual.
- MSE = relative metric that calculates the loss. smaller the value the better.
- RMSE = gives absolute metric in the same unit as the label. smaller the value the better. In essense, it represents the average number of "label" or "target" by which the predictions are wrong!
- R^2 = a relative metric, higher the better.
  In essence, it represents how much of the variance between predicted and actual values the model is able to explain.

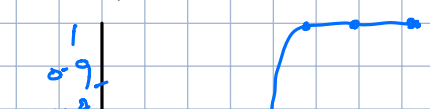
Improve the model?
- we did OLS, we can try Lasso and Ridge.
- Tree-based algorithms. (Dtree Regressor)
- Ensemble algorithms.
  - RandomForest Regressor (Bagging)
  - GB Regressor. (Boosting) (wins)

↙
Tune the hyperparameters - Grid search CV.

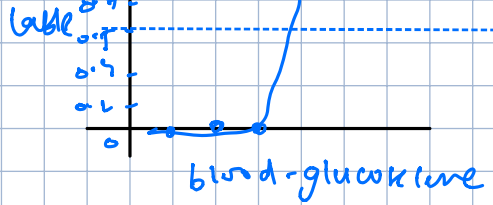
- Now build the models using normalized and encoded features.
  normalization - MinMax, Z-Score
  encoding - ordinal encoding, one-hot encoding.
- pipeline function in sklearn.
- StandardScaler() = Z-score.

- Dump the best model as 'pkl' and use it to predict new values.

3. Train and evaluate classification models.
(Theory)
- model to predict the probability of each "class"
- logistic function
- confusion matrix for evaluation



- Accuracy
- Recall
- Precision.



(Jupyter)

- diabetes patient use case
- plot box-plot of features grouped by "target"
- fit logistic with regularization parameter.
- plot accuracy, precision, recall, f1-score.

- what gets predicted is the probability that the label is true. and not the actual label. Threshold of 0.5 is used to then identify the actual label.

- Changing the threshold will change everything.
- plot ROC for different thresholds.
- calculate AUROC, closer to 1, the better.
- Use sklearn pipeline with encoded features.
- Try different algorithm.
  - SVM
  - Tree based
  - Ensemble. RandomForestClassifier

- Multiclass classification.
  - One over Rest (OVR)
  - One over one (OVO)

- perform similar steps for multiclass and dump the model as pkl.

4. Train and evaluate clustering models.

(Theory)

- no "ground truth" available.
- labels are which cluster the observation belongs to.
- K-means working algorithm? write pseudo-code?
- Clusters are identified based on similarities/distances in the n-dimensional place.

(Jupyter)

- seeds data. 6 dimensions / features.
- use PCA to reduce to 2 dimensions. Normalize before applying PCA.
- Visualize the 2D data in scatter plot
- How to know how many clusters?
- Use different cluster, calculate within cluster sum of squares (wcss) measures the tightness. low wcss = data points are closer.
  Plot wcss for each model.
- This creates an "elbow" like shape.
- Clustering can be used as an initial step towards classification. Use cluster numbers as labels.

- Hierarchical clustering.

↳ divisive / Top down
↳ Agglomerative / bottom up.
L1 distance = Manhattan distance
L2 distance = Euclidean distance
- cosine similarity.

- Data Integration
- model Build (have complete control over the choice of languages)
- model monitoring, APIs, deployment.

- choice of languages / interfaces.

5. Train and evaluate deep learning models.

(Theory) DNN

- strong roots in neuroscience
- $f(x, w, b)$
- activation function decides whether to activate a neuron.
- input layer, hidden layer, output layer.
- multilayer perceptron, fully connected networks.
- Training - epochs, batches, mini-batches.
  ↳ pass to neurons in feed-forward manner. variance / loss is calculated.
  - adjustments are back-propagated to minimize the loss.
  - Then next epoch / batch is run.
- why process training features in batch?
- How loss is calculated?
- How to optimize / reduce the loss?
  - NNs are in essence one big nested function.
  - derivate / gradient helps in finding out whether loss is increasing / decreasing for a given weight / bias.
  - this is the job of an optimization algorithm? SGD, ADADELTA, Adam.
- By how much should the optimizer adjust bias / weight values?
  ↳ learning rate decides this.

(Jupyter) - PyTorch
- Penguin dataset. DNN works well if features are on similar scale.
- Explore, sklearn train_test_split.
- Install Pytorch, convert data to tensor objects.
- After each epoch, the validation set is passed to compare the performance using data on which is not trained.
- The 'loss' of training data is passed to the optimizer → applies partial derivatives and determines the direction → whether to increase / decrease the weights / bias.
- how much to adjust = learning rate
- Backpropagate the weights & biases. Then the next epoch begins.

2 things we are looking at -
- loss should reduce after each epoch → model is learning.
- Training & validation loss should follow similar trend → model is not overfitting.

- use sklearn confusion matrix.
- Save model as .pt file.
- For tensorflow, similar steps, save as .h5 file.

5.b) Convolutional Neural Networks (Theory) — CNN

- convolution, filters (kernel) = matrix of weight values.
- kernel is convolved across the image, to get feature maps, or extracted features of an image.
- pooling / downsampling e.g. - max-pooling. kernel is used here also.
- dropping layers effective method to prevent overfitting.
- flattening the values and end with FCN.
- same concept of batches, epochs, backpropagation is applied here.
- what is adjusted here? filter kernel weights, weights / biases in FCN.

(Jupyter)

- Many points same as above.
- keras ImageDataGenerator.
- Training shows ETA ∈ Good feature.

5.c) Transfer Learning (Theory) — Transfer Learning

- It is often easier to learn a new skill if you have expertise in a similar, transferable skill. easier to teach someone how to drive a bus if they know how to drive a car.

(Jupyter)

- Take the learned weights of feature extraction layers of one model and apply to your problem.
- A professional tennis player and a novice. It's easier to teach the professional basics of "racquetball" as some of the underlying principles are already learned.
- Many models available - Resnet, VGGNet etc.