Problem Statement

I am given dataset and my task is to predict labels for the testing data. This assignment is not a case study. To save space, I will not add results that are already given in notebook and .pdf file. I am not given domain, feature details or end result implications I take following steps to solve this problem.

Data Preprocessing

- After loading using pandas, I check if there are any *missing values* in data. There is none.
- Next, I check label data in training set. I find that categorical classes are i*mbalanced* i.e 89.9% for 1 and 10.08% for -1. I also change -1 with 0 for my own comfort. Minus signs give me negative vibe.
- I *normalize* all features because I notice that some features have very high values and some have very low values. In econometrics, we create plots to check outliers. As I am not sure what this data is about and how much high or low value is an outlier threshold therefore, I keep it simple. I use standardScaler for normalization.
- Finally, I apply dimension reduction method. I use PCA to see if reducing 10,000 dimension would give any better results. I keep components with 95% variance.

Model Building with machine learning algorithms

For model building, I use two methods for experiment. I use train-test split by keeping 80-20% ratio for train-test data. One thing to clarify that test is more like validation data for my experiment because test data is already given. Secondly, I use K-fold with cross validation of 10 folds. I use multiple machine learning algorthms provided from sklearn module. I make sure that I am feeding an array of features to my model. Aim is to see how each model performs on 10,000 dimension and over pca component features.

With 10,000 dimensions

I already have attributed X as features i.e 10K and y as label. Further split into train-test set created X_train, X_test, y_train and y_test .Results with k-fold are given

Models	Accuracy
Linear Reg. Classifier	89.7%
Naïve Bayes	89.9%
K-Nearest Neighbour	86.8%
Random Forest	89.9%
Support vector Classifier	89.9%
Decision Tree	93.8%

As Decision tree shows best accuracy result therefore, I test it further. I check using train-test split to find how well it performs on training and validation set. It gives 100% accuracy on training data and 93.8% on validation. It is classical case of overfitting. Though our model gives good performance score but, it is learning way too well and hence it would fail to generalize on new unseen data.

With PCA

For PCA, I use explained variance ratio of 94.21 and I use 3000 component dimensions. I get following results using k-fold.

Models	Accuracy
Linear Reg. Classifier	87.9%
Naïve Bayes	89.8%
K-Nearest Neighbour	87%
Random Forest	89.9%
Support vector Classifier	89.9%
Decision Tree	80%

Again, I use train-test experiment with Random Forest(with highest accuracy). I again get overfitting. There are different methods to keep overfitting in check. One is to tune model which I did. Other problem that I notice is in classification report.

	Precision	Recall	F1
0	0.33	0.01	0.02
1	0.89	1.00	0.94
accuracy			0.88
Macro avg	0.61	0.50	0.48
Weighted avg	0.82	0.88	0.83

This clearly shows that class imbalance is a problem for given case. Class type 0 i.e -1 in our data is not occurring that frequent and hence, model has not learnt to predict it that well. This is also a

reason why accuracy is not a good evaluation metric. In such cases, F1 is taken as evaluation criteria. I apply fully connected dense neural network to see if it could provide better results.

Deep learning model

There is no structure in given data as opposed to text or image data therefore, I shall implement fully connected dense neural network. For this, I use same train-test split experiment method, use normalized data, and create encoded labels as of 0 and 1.I use 10000 dimension as I could see that PCA did not help much in improving performance of model and overfitting problem. Details of neural network model is given in notebook. I will highlight some of key points:

- I have used “stacked” hidden layer with feedforward network settings.
- Between each layer, I use “rectified linear unit (relu)” i.e if input > 0: return input else: return 0
- To avoid overfitting, I use dropout. I also use “BatchNormalization” to make artificial neural networks faster and more stable through normalization of the input layer by re-centering and re-scaling.
- For output layer (having two possible outcomes), I use “softmax” to keep probability distribution ranging from 0.0-1.0. If it is greater than 0.5 then it belongs to class 1 else class 0.
- For backpropagation, I use “ADAM” as optimizer. There are other options such SGD, RMSProp etc. I find that adam works best for back propagation for fully connected dense neural network. adam uses combination of learning rate and momentum.
- loss is “binary crossentropy” as I have expected two outcomes (0 or 1).
- “EarlyStopping” is used to control extra computational wastage.
- “class weight” is used since we find an imbalanced class case.

Evaluation and performance

As for results, we get 94.1% accuracy and loss score of 0.21 which is highest so far. However, there is some overfitting. This is not as high as was in machine learning model. It would be more helpful if I have more information about data reliability and its source. Additionally, I am unable to do sanity check on this data because I do not get any domain specific information from description. Still, I think this is a good model. I have this conclusion because of classification

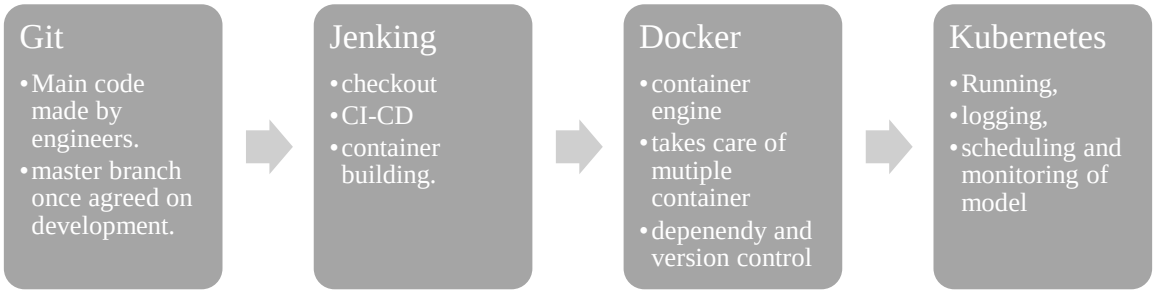
report results where there is more consistency in accuracy, precision, recall and F1 score between two classes. Unlike machine learning model, neural net model performs better with low representation class i.e 0.

	Precision	**Recall**	**F1**
0	0.70	0.86	0.77
1	0.98	0.95	0.97
accuracy			0.94
Macro avg	0.84	0.91	0.87
Weighted avg	0.95	0.94	0.94

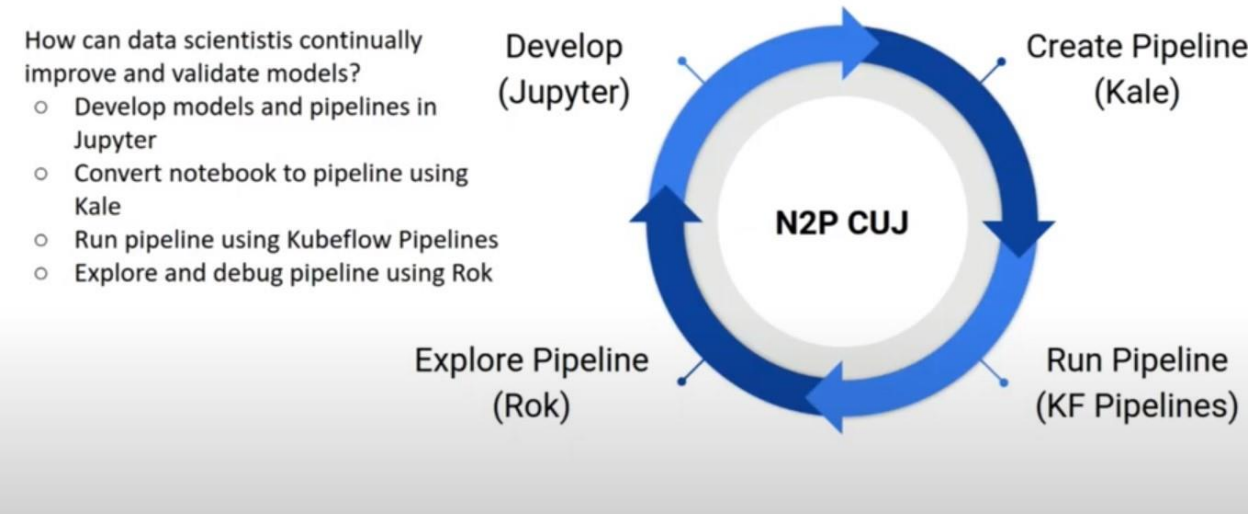
- Some of practices made are checking model weight, distribution of weights across neural network with each stacked layer. I find weight distribution is "bell shaped" which shows that model is consistent across all hidden layers.
- Predict Proba shows probability distribution between 0 and 1 i.e 0 being lowest and 1 being highest. Standard threshold as per softmax is 0.5. If I am asked to improve results for a specific class type then I need to tweak threshold of probability lower or higher. For example, if I decrease this threshold then I could get more precise results for class 0 i.e class -1 of our original data. ROC and AUC curve confirms this point and I use them for further verification even though I am not specifically asked to take care of imbalanced classes.

Model Deployment to production

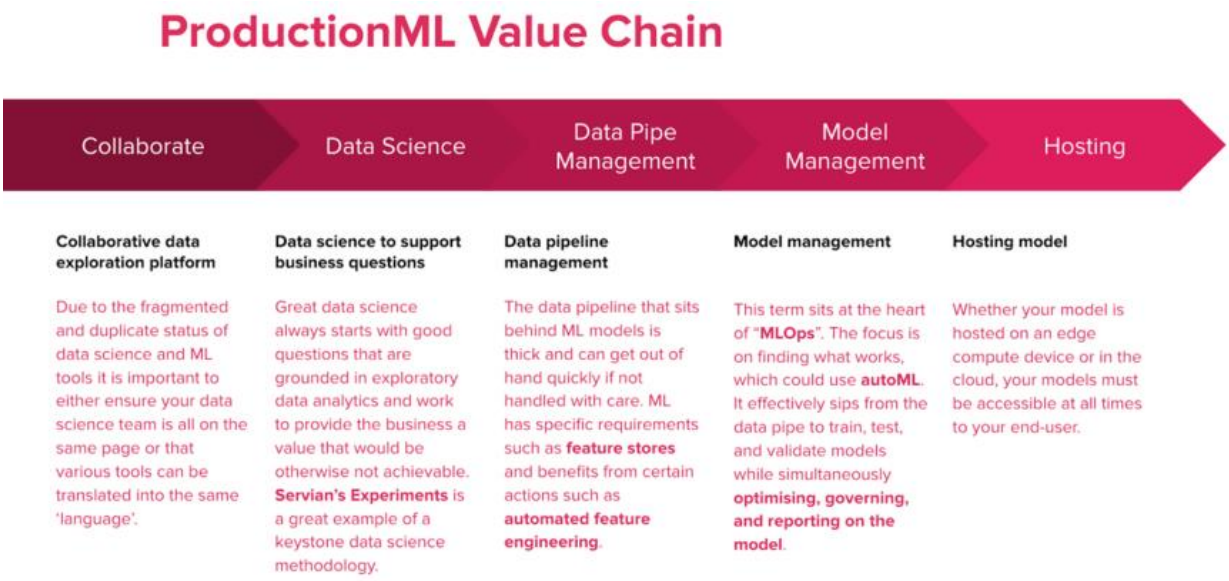
In my opinion, models are only alive when they are used in production. Model deployment part is crucial for ML projects. I am not asked to perform that but, I would like to provide a conceptual framework with methods and tools for model deployment. From submitted code, one could create this whole workflow.



As one could see that this approach works better on team projects than individual projects. In simple projects like this, one may only be more concerned with performance however, in real projects it matters. There are other tools that are really interesting such as MLflow and Kubeflow. These tools enable CI-CD in machine learning domain. This is shown in one of MLOps talks that recently attended.



Another interesting concept is “Production value chain”. I think these tools and approaches could make a systematic ML pipeline.



Source given in references

Submission

I am submitting following files:

a. Jupyter notebook: It contains code and it can be run given all modules
b. PDF version of notebook contains all results and my comments as well. Sometimes running whole code and waiting for results is boring so, I am providing this file.
c. test_label.csv: shows all predicted values as 1 & -1
d. This report document
e. Requirement.txt: contains all key modules used in this code

References

https://medium.com/weareservian/the-cheesy-analogy-of-mlflow-and-kubeflow-715a45580fbe