CS 725 : Foundations of machine learning

Analysing and Boosting of ML models for Diabetes prediction

Team members & Work Done

ROLL NUMBER	NAME	WORK DONE IN PROJECT
22M0750	ABHINAV GARG	Trained & tuned Neural network classifier, Implemented Boosting.
22M0761	ABHISHEK DIXIT	Trained & tuned Logistic regression classifier, Studied and Tested scaling methods.
22M0770	NIKHIL SHARMA	Trained & tuned Decision tree, Implemented Bagging (Random forest classifier), Studied and Tested under-sampling methods.
22M0780	SHUBH MALHOTRA	Trained & tuned Naive bayes classifier, Implemented Boosting.
22M0747	BALBIR SINGH	Trained & tuned Support Vector Machine classifier, Studied and Tested over-sampling methods.

Introduction & Problem Statement

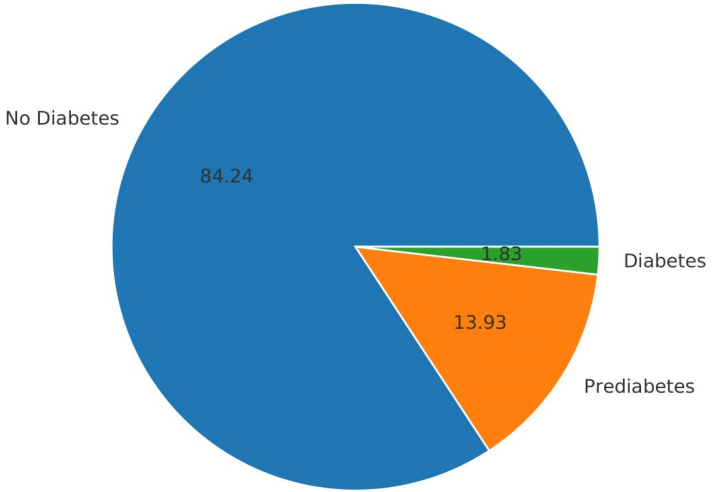
- One in every eleven people in India suffers from diabetes. In 2020, around 700,000 Indians died of diabetes or other complications caused by diabetes.
- 10.5 % of the world's population suffers from some form of diabetes.
- Early detection of Diabetes is necessary to counter it.
- It is very hard to find the actual cause of diabetes and hence **prediction** based on multiple factors is a good approach.
- We have taken a dataset [1] with 21 input features including BP, Cholesterol, BMI, Smoking, Physical activity, Mental health etc. and 1 output value which tells if the person is suffering from Diabetes or not.
- We have implemented a **Hybrid Ensemble** using multiple classifiers trained by us to obtain a combined accuracy more than the individual models, to predict the target variable of **Diabetes** positive or negative in a person.

Summary of the Tasks performed

1. Exploratory Data Analysis to understand the composition and behaviour of the input and output variables.
2. Performed Data preprocessing to make the data suitable for training and prediction.
3. Performed further Data preprocessing to make the data optimal for individual models.
4. Trained Neural network, Logistic regression, Decision tree, Random forest, Naive bayes and SVM classifiers.
5. Implemented grid search for hyperparameter tuning of all the classifiers.
6. Implemented hybrid ensemble method i.e. Boosting to increase the collaborative accuracy of the models as compared to the individual models.

Description of the Dataset

- The dataset is in **.csv** format and was acquired from Kaggle [2]. The *Centers for Disease Control and Prevention (CDS)* in America conducts a telephony survey every year and the Dataset used by us was collected in 2015 under the *The Behavioral Risk Factor Surveillance System* survey.
- The dataset originally has 253680 rows containing the survey data collected from individuals.
- We have chosen this dataset as it contains information from people of different age, education, income and other social determinants of health.
- The dataset contains **21 input features** which include various details about the individuals.
- The dataset has 1 target variable Diabetes_012 which has 3 classes. 0 is for no diabetes, 1 is for prediabetes, and 2 is for diabetes.
- By doing **Exploratory Data Analysis**, we found out that the dataset is **highly skewed** with respect to the output variables. The distribution is shown here -



Main techniques used

Sources of code used & Functionalities implemented

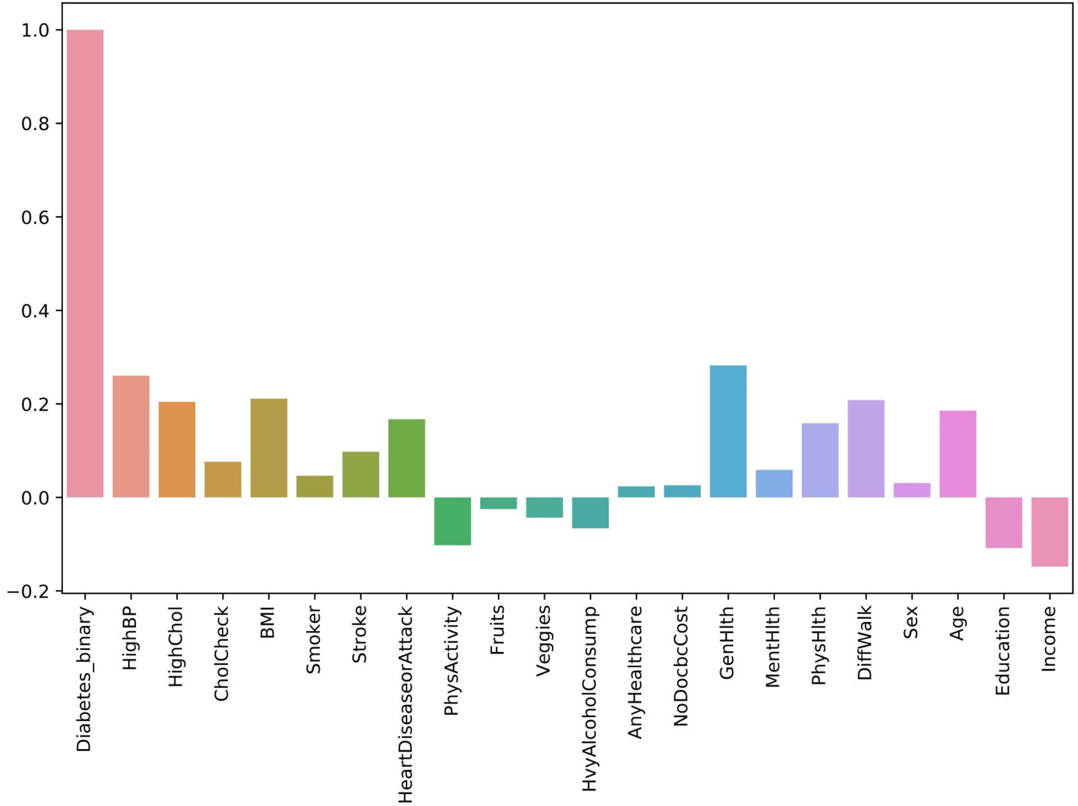
- Implemented the following functionalities -
  - Transforming multi class target variable to binary variable.
  - Implemented functionality to identify and convert categorical input variables to multiple columns using one hot encoding.
  - Implemented Random forest classifier.
  - Implemented grid search using for loops for finding the most optimal hyperparameters.
  - Implemented Boosting for Hybrid Ensemble of all the models.

- Libraries were used for the following functionalities -
  - “numpy, pandas, matplotlib, seaborn, train_test_split” for their respective basic functionalities.
  - “imblearn.under_sampling” for undersampling and “imblearn.over_sampling” for oversampling the data.
  - “sklearn.preprocessing” for various scaling methods.
  - “sklearn” for Logistic regression, Naive Bayes, Decision Tree, SVM.
  - “keras” for Neural Network model.

General data preprocessing

1. First of all, we dropped the duplicate rows as they increase the training time of the model and also do not significantly contribute to the accuracy of the model.
2. **Number of rows changed 253680 - - > 229781**
3. As the data has a highly skewed distribution between the classes of the target variable, we decided to **combine the 2 classes of Prediabetes (1) and Diabetes (2)**, as doing undersampling on 3 classes would have reduced the total number of rows to a very small number.
4. Distribution of the output variable changed like -
5. **Initial =>** 0 = 190055, 1 = 4629, 2 = 35097
6. **Final =>** 0 = 190055, 1 = 39726 (The multi-class classification problem now got converted to a binary classification problem)
7. Plotted a correlation plot (next slide) between the input variables and target variables and dropped the variables with near 0 correlation with the target variable. This decreased the dimensionality of data while not having a significant impact on the accuracy of the models.
8. **Dropped variables - Fruits, Veggies, AnyHealthcare, NoDocbcCost, Sex.**

Correlation Plot



Naive Bayes Classifier

1. Implemented Gaussian Naive Bayes where likelihood of features is assumed to be Gaussian

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

2. Plotted correlation between all the features.
3. Dropped the features with very high correlation as Naive Bayes has an assumption of conditional independence between input features as they are likely to be dependent on each other.
4. Features Dropped - ('PhysHlth', 'Income', 'DiffWalk', 'MentHlth')
5. Tuned the hyperparameter for smoothing factor and found the optimal value to be 2e-9.

ACCURACY ACHIEVED = Train: 0.783755, Test: 0.782114

Logistic Regression

1. Converted categorical & continuous input variables into multiple columns using One Hot encoding.
2. Tested various undersampling and oversampling techniques and finally settled with SMOTE
3. Tested various scaling methods and found MinMaxScaler to be optimal
4. Implemented grid search to find optimum values of the following hyperparameters -
   4.1. **penalty =** {'l1', 'l2', 'elasticnet', 'none'}, default='l2'
   4.2. **tol =** float, default=1e-4
   4.3. **C =** float, default=1.0
   4.4. **solver =** {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'
   4.5. **max_iter =** int, default=100
   4.6. **l1_ratio =** float, default=None
5. Found the following optimal values of hyperparameters -
   5.1. penalty = elasticnet
   5.2. tol = 1e-3
   5.3. C = 0.9
   5.4. solver = saga
   5.5. max_iter = 300
   5.6. l1_ratio = 0.9

ACCURACY ACHIEVED = Train: 0.871671, Test: 0.872690

Decision Tree Classifier

1. Decision Trees is a very simple supervised machine learning algorithm. It works similarly to how we humans make decisions in daily life.
2. On every split or decision we want to choose the best feature using a splitting criterion. I have used Gini index for that.
   Why gini?
   - Gave same accuracy as that of entropy.
   - Is faster than the entropy

3. Optimal depth found: 13 out of the 51 features
4. Resampling techniques like undersampling and oversampling was used to balance out the data. Decided to use undersampling. Why?
   - Was getting same accuracy as that of over sampling
   - Smaller size of dataset, hence less time taken to train the model.

ACCURACY ACHIEVED = Train: 0.843849, Test: 0.828996

Random Forest Classifier

1. RandomForest is an ensemble of decision trees.
   Why random forest?
   - Decision trees have high variance
   - It reduces overfitting of decision trees using a technique known as bagging
2. Bagging
   - Bootstrapping:
     1) Dataset for each tree is created by randomly selecting the data from the original dataset with replacement.
     2) Each tree uses only a random subset of features.
     3) This reduces correlation between the trees.
   - Aggregation: Class with the majority of the votes from all the models will be chosen as the final prediction.
3. Random forest has been implemented using the Decision tree library from the sklearn as a subroutine.
4. Each tree has been trained till the 0.2 of the max depth.

ACCURACY ACHIEVED = Train: 0.840585, Test: 0.836927

Support Vector Machine (SVM)

Preprocessing Steps:

1. Different Feature scaling method applied i.e. MaxAbsScaler, MinMaxScaler, StandardScaler. Out of which **MaxAbsScaler** was found optimal.
2. Undersampling - Tried different Undersampling Method, Best one is **NearMiss**.
3. Converting categorical features into multiple features using One-hot encoding. (except for features : ‘BMI’ , ‘MentHlth’, ’PhysHlth’, ‘Age’)

Linear Kernel, Polynomial Kernel, Radial Basis Function (RBF) Kernel, these three kernel tricks when used with optimal value of Hyperparameter, **RBF** gives better performance.

Grid search is implemented to tune the following Hyperparameters:

- **C:** C is the tuning parameter that decides how much importance is given to error or slack variables. A large C gives you low bias and high variance.
- **gamma**: gamma is the parameter of RBF to handle non-linear classification.
- **degree**: degree of the polynomial kernel functions, It is only applicable to the polynomial kernel and not applicable to other kernels.
- **tol**: error tolerance, default values is 0.001

Following are the best values of hyperparameter for our dataset:

- Optimal kernel - RBF
- Optimal C = 20
- Optimal gamma = 1
- Optimal tol : 0.001

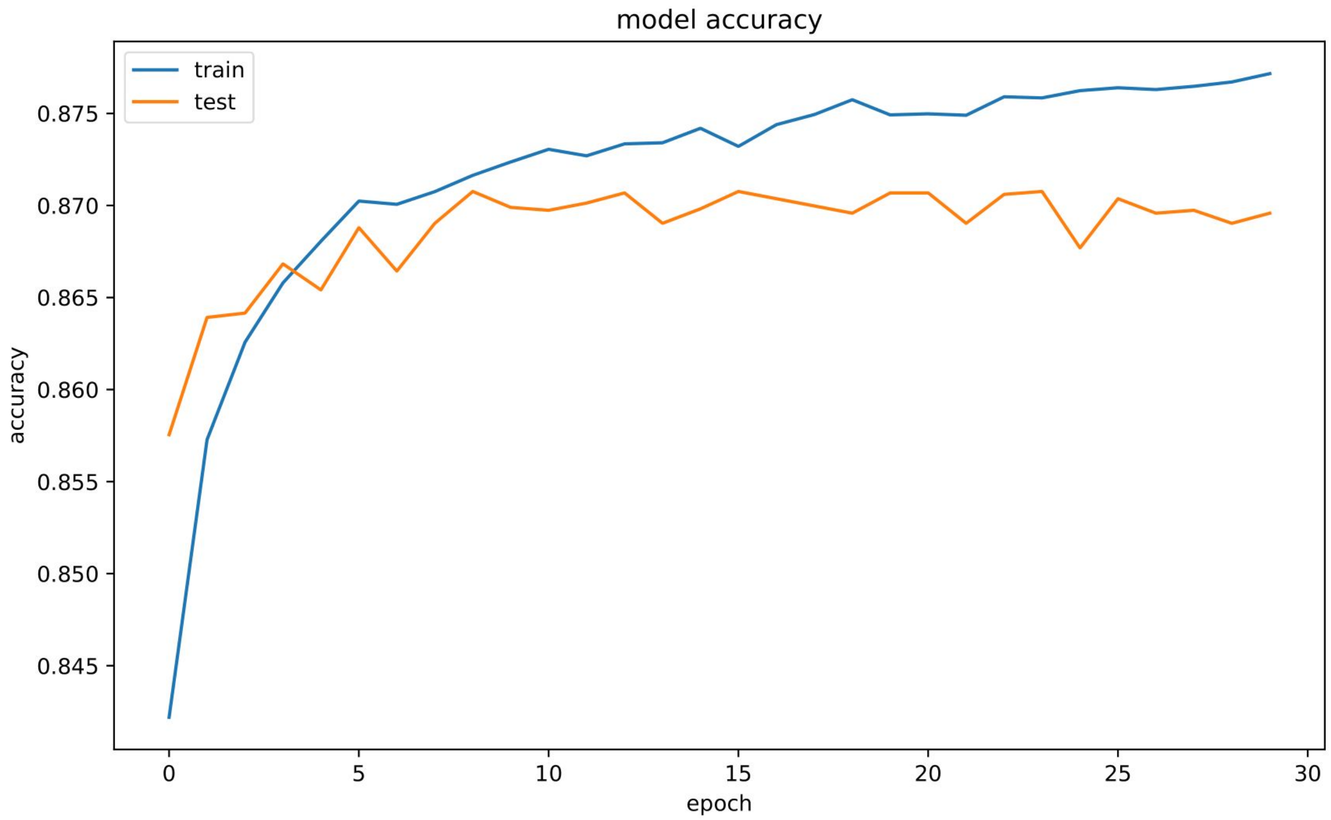
ACCURACY ACHIEVED = Train: 0.913909, Test: 0.897175

Neural Network

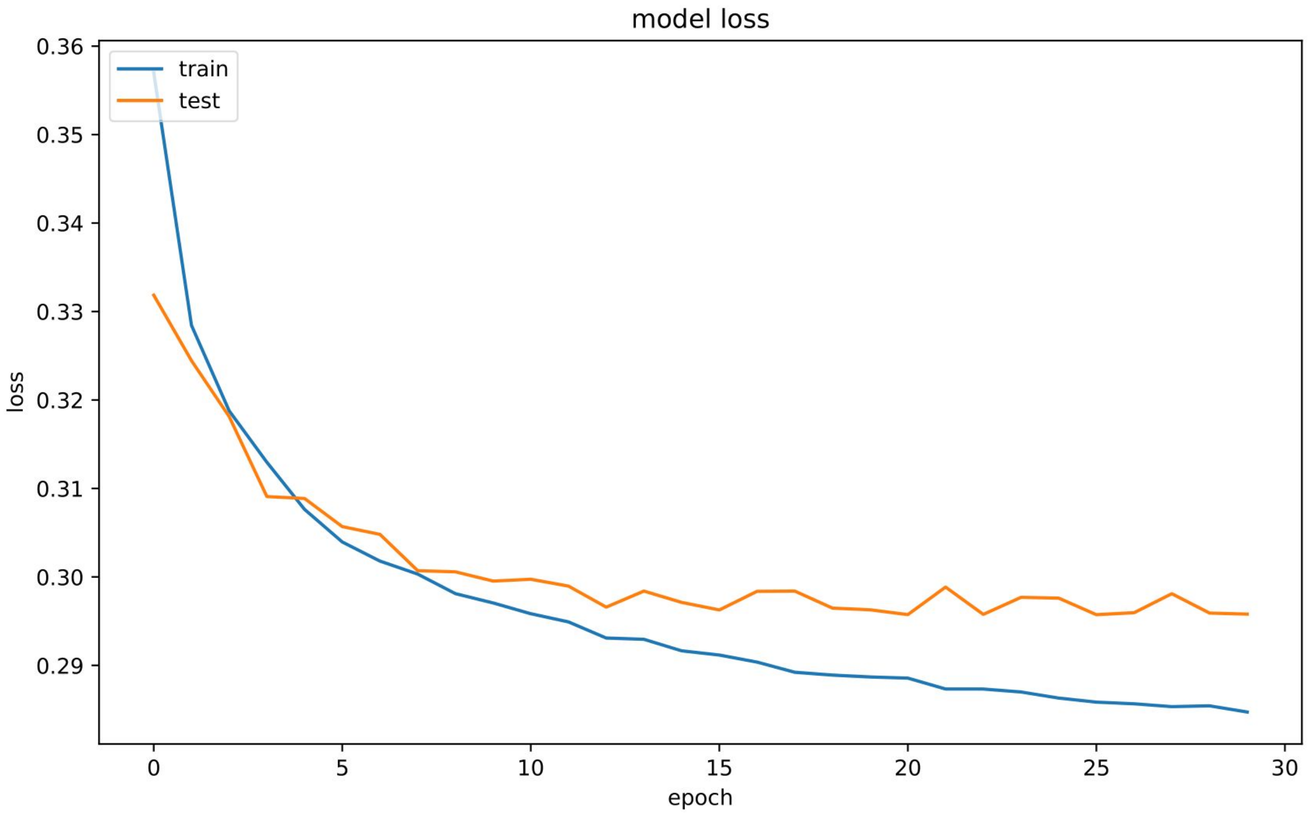
- Tested various undersampling and oversampling techniques and finally settled with NearMiss(n_neighbors = 20)
- Scaled the data using RobustScaler
- Implemented grid search for tuning the following hyperparameters -
  - Batch Size
  - Learning Rate
- Trained neural network with 2 hidden layers using **relu** as activation function.
- The architecture for the NN is - “ X_train.shape[1], 32, 64, 1 “
- Used Sigmoid as activation function at output layer
- Used binary cross entropy loss as the loss function.

ACCURACY ACHIEVED = Train: 0.877330, Test: 0.870367

Neural Network model Accuracy graph



Neural Network model Loss graph



Boosting

- Boosting methods build ensemble model in an incremental way.
- The main principle is to build the model incrementally by training each base model estimator sequentially.
- The idea of boosting is to basically combine several weak classifiers which are sequentially trained over multiple iterations of training data.
- For each weak classifier, repeat the following :
    - Calculate the error as sum of weight of all misclassified points.
    - Calculate alpha that determines the importance given to that classifier while testing.
    - Update weights of all training points to give more importance to currently misclassified points.
- The final accuracy obtained is generally better than the accuracy of the individual constituent models.

ACCURACY ACHIEVED = Train: 0.862324, Test: 0.851947

Results obtained (Using accuracy as a metric)

Model Name	Initial test accuracy (without preprocessing and hyperparameter tuning)	Final Accuracy (after preprocessing, grid search and hyperparameter tuning)
Naive Bayes	0.753587	Train: 0.783755, Test: 0.782114
Logistic Regression	0.833040	Train: 0.871671, Test: 0.872690
Support Vector Machine	0.827803	Train: 0.913909, Test: 0.897175
Decision Tree	0.779932	Train: 0.843849, Test: 0.828996
Random Forest	0.842925	Train: 0.840585, Test: 0.836927
Neural Network	0.830689	Train: 0.877330, Test: 0.870367
Final Model (Hybrid Ensemble)	-	Train: 0.862324, Test: 0.851947

Result of Boosting (Hybrid Ensemble)

- SVM Training Accuracy: 0.8554193037974683
- Neural Network Accuracy : 0.739661248561565
- Logistic Regression Training Accuracy: 0.804067174913694
- Decision Tree Training Accuracy: 0.5716160817031071
- Naive Bayes Training Accuracy: 0.8117628020713463

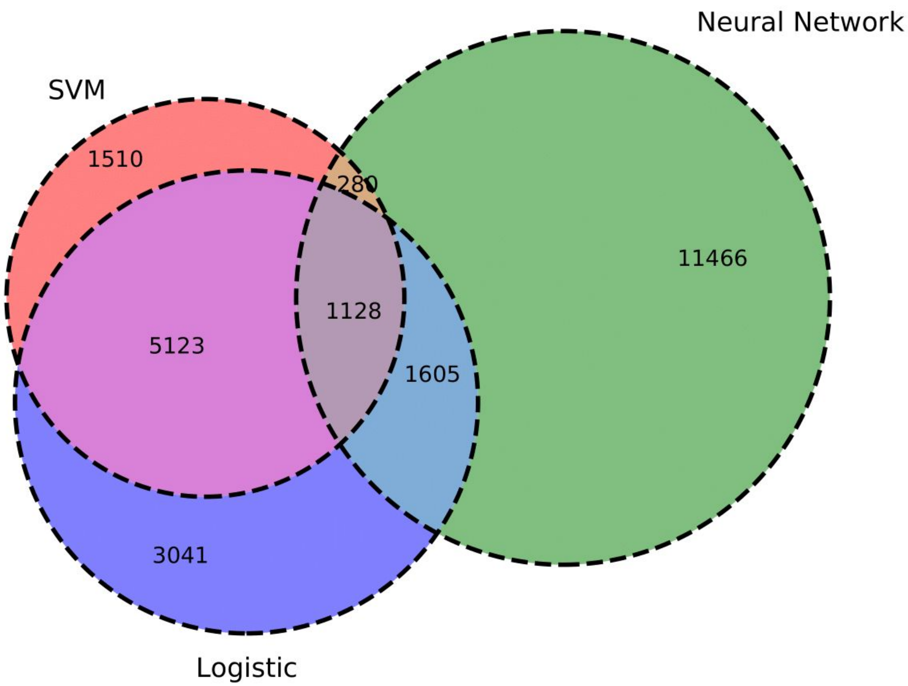
- **Training accuracy obtained after Hybrid Ensemble - 0.862324**

Why individual models accuracy less than before?

1. We are giving higher weight to misclassified points, hence models are struggling to fit them.
2. We used common data preprocessing for all the models unlike specific for each one.

Why not drastic improvement between individual models and final hybrid ensemble classifier?

1. Models were already strong classifiers, hence not possible to increase accuracy and reduce errors above a certain limit.
2. Large overlap is present in the misclassified data points, hence many data points are misclassified by every classifiers, as indicated by the venn diagram for 3 of the classifiers. (no library support for more than 3 element venn diagram)



Links for the project code -

- GitHub - https://github.com/Abhishek-Cypher/FML_22M0747_22M0750_22M0761_22M0770_22M0780

- Google Colaboratory - https://colab.research.google.com/drive/18n_X3YoaYcKPqyiBE4NyoctUHl3IxopG?usp=sharing