MACHINE LEARNING MODEL TO PREDICT BIRTH WEIGHT OF NEW BORN USING TENSORFLOW

S.Karthiga, K.Indira and C.V.Nisha Angeline

Assistant Professors, Department of Information Technology, Thiagrajar College of Engineering,Madurai

ABSTRACT

Low Birth Weight is the major problem for the new born. Low birth weight is a term used to describe babies who are born weighing less than 5 pounds, 8 ounces (2,500 grams). Low-birth weight babies are more likely than babies with normal weight to have health problems as a newborn. Almost 40 percent of the new born suffer from underweight. Predicting birth weight before the birth of the baby is the best way to help the baby get special care as early as possible. It helps us to arrange for doctors and special facilities before the baby is born. There are several factors that affect the birth weight. Through past studies, it has been observed that the factors which affect the child birth range from biological characteristics like the baby's sex, race, age of mother and father, weight gained by the mother during pregnancy to behavioral characteristics like smoking and drinking habits of the mother, the education and living conditions of the parents. This project focuses on developing a web application that predicts baby weight taking baby's gender, plurality, gestation weeks and mothers age as inputs. Machine learning is one of the domains that plays important role in medical industry. Many machine learning models have been developed to predict diseases at the early stage. In this project wide and deep neural network model is developed using TensorFlow library in Google cloud environment. Wide and Deep Neural Network combines wide linear model and deep neural network. It provides both memorization and generalization. Pre-processing and training is done in the distributed environment using cloud Dataflow and Cloud ML Engine. The model is then deployed as REST API.A web application is developed to invoke the API with the user inputs and show the predicted baby weight to the users. It is scalable and provides high performance.

1. INTRODUCTION

The aim of the project is to predict the baby weight so that the baby can get better care. It is done using machine learning model and in cloud environment. Machine learning plays a major role in medical diagnostics. Machine learning in medicine has recently made headlines. Google hasdeveloped a machine learning algorithm tohelp identifycancerous tumours on mammograms. Algorithms can provide immediate benefit to disciplines with processes that are reproducible or standardized. Machine learning can offer an objective opinion to improve efficiency, reliability, and accuracy. We'll be able to incorporate bigger sets of data that can be analyzed and compared in real time to provide all kinds of information to the provider and patient. This project uses Wide and Deep Neural Network model which provide both generalisation and memorization. Cloud augments machine learning by enabling process in distributed environment providing benefits of

N. Ramasubramanian et al.(Eds) : SRAIC-2019
pp. 71-90, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.91506

scalability,high performance, availability, maintainability, repeatability, abstraction and testability.

Problem Statement

A survey says almost 40 percent of new born suffer from underweight. An underweight baby is more likely susceptible to many health problems than a baby with normal weight. Special care and facilities are to be given to them when they born so that their health condition will be improved. Predicting the weight before the birth of the baby and arranging doctors and facilities if the weight of the baby is less than 5 pounds and 8 ounces helps the baby get better treatment. The solution is to create a machine learning model that predicts the baby weight from major factors like gestation weeks, mother's age, gender and plurality of baby. The model should be deployed as REST API so that it can be invoked using a web application to get predicted birth weight. The process should be carried in cloud environment so that it will be auto scalable and provides high performance.

Objectives

To collect and analyze the natality dataset from big query.

- To launch a preprocessing pipeline using cloud Dataflow to create training and evaluation datasets.
- To create Wide and Deep Neural Network Modelusing TensorFlow and train model in AI Platform.
- To deploy model as REST API.
- To create a web application that invokes API to predict the birth weight.

The scope of the project is to develop a web application that takes mother age, gestation weeks, plurality and gender of the baby as input and give birth weight as output. The application is highly helpful to mother and hospital to make facilities to care of the baby before its birth. If a mother is on the way to the hospital, she calls the nurse. This work is implemented in Deep netural network because of the following reasons: Deep Learning is the next generation of machine learning algorithms that use multiple layers to progressively extract higher level features (or understanding) from raw input. Deep learning algorithms are now used by computer vision systems, speech recognition systems, natural language processing systems, audio recognition systems, bioinformatics systems and medical image analysis systems and the special features of Deep learning is

- No need for feature Engineering
- Best results with unstructured data
- No need for Labeling Data

2. RELATED WORK

2.1. Wide and Deep Learning for Recommendation System

This paper presents Wide & Deep learning jointly trained wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. It productionized and evaluated the system on Google Play, a commercial mobile app store with over one billion active users and over one million apps. Online experiment results show that Wide & Deep significantly increased app acquisitions compared with wide-only and deep-only models. A recommender system can be viewed as a search ranking system, where the input query is a set of user and contextual information, and the output is a ranked list of items. Given a query, the recommendation task is to find the relevant items in a database and then rank the items based on certain objectives, such as clicks or purchases. During training, input layer takes in training data and vocabularies and generate sparse and dense features together with a label. The wide component consists of the cross-product transformation of user installed apps and impression apps. For the deep part of the model, A 32 dimensional embedding vector is learned for each categorical feature. They concatenate all the embeddings together with the dense features, resulting in a dense vector of approximately 1200 dimensions. The concatenated vector is then fed into 3 ReLU layers, and finally the logistic output unit. The Wide & Deep models are trained on over 500 billion examples. Every time a new set of training data arrives, the model needs to be re-trained. However, retraining from scratch every time is computationally expensive and delays the time from data arrival to serving an updated model. To tackle this challenge, we implemented a warm-starting system which initializes a new model with the embeddings and the linear model weights from the previous model. Before loading the models into the model servers, a dry run of the model is done to make sure that it does not cause problems in serving live traffic. We empirically validate the model quality against the previous model as a sanity check.

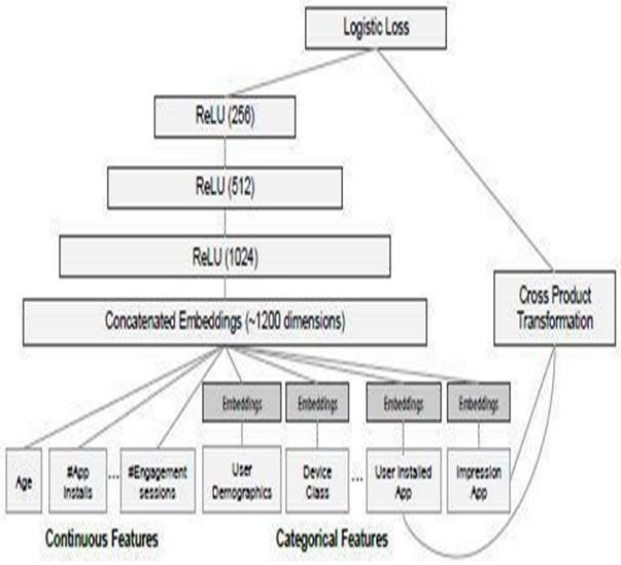

Fig: 1 Wide and Deep model structure for apps recommendation

2.2. Prediction and Classification of Low Birth Weight Data using Machine Techniques

The objective of this research was to apply one of the ML techniques on the low birth weight (LBW) data in Indonesia. This research conducts two ML tasks, including prediction and classification. The binary logistic regression model wasfirstly employed on the train and the test data. Then, the random approach was also applied to the data set. The results showed that the bi-nary logistic regression had a good performance for prediction, but it was a poor approach for classification. On the other hand, random forest approach has a very good performance for both prediction and classification of the LBW data set. Binary logistic regression is a type of logistic regression, which has only two categories of outcomes. It is the simplest type of logistic regression. The main goal of binary logistic regression is to find the formula of the relationship between dependent variable *Y* and predictor *X*. Random forests are defined as the combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. In this research, the LBW data were obtained from the result of 2012 IDHS. In the beginning, the raw data consists of 45607 women aged 15-49 years as the respondents. After data cleaning process, the amount data reduced to 12055 women aged 15-49 years who give birth from 2007 up to 2012. The dependent variable is Low Birth Weight with two categories. The independent variables are Place of Residence, Time Zone, Wealth Index, Mother's and Father's Education, Age of the mother, Job of the mother, Number of children.

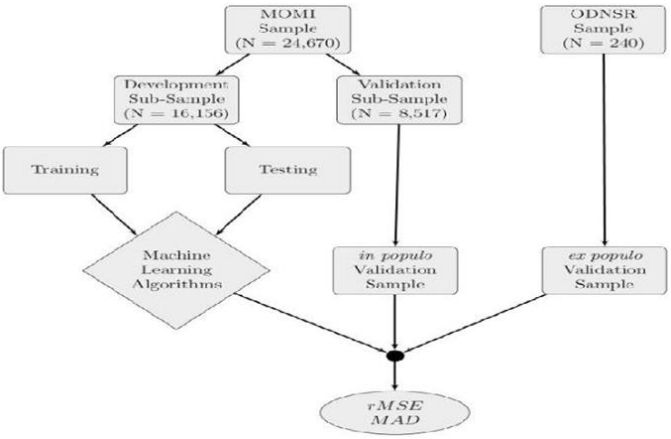

Fig:2 Strategy to predict fetal growth

rMSE is root Mean Squared Error, MAD is median absolute deviation, MOMI is Magee Obstetrics Maternal and Infant Database, ODNSR is Obstetrical Determinants of Neonatal Survival Study Data.

2.3. Prediction of Birth Weight of a Baby

The Dataset used in this research is NC VITAL STATISTICS BIRTHS Dataset.For the year 2008, it had 133422 rows with 125 columns. They trimmed it down by removing the features

which are not relevant. We have also refined the dataset by splitting the categorical variables into several binary variables. After all the cleansing, the final dataset has 69051 rows and 52 columns. The dataset was divided into training and test data in a random manner. The training set comprises of 85% of the data. Whereas, the test dataset comprises of the remaining 15%. It is observed that Boys are found to weigh slightly more than girls at birth, birth weight of a baby born to a smoker mother is less than that born to a non-smoker mother. Whereas, the mortality rate for LBW babies born to a smoker mother is less, babies born to black parents weigh more than those born to white parents. Also, babies born to black father and white mother are a little heavier compared to one born to a white father and black mother, birth weight of a baby born to a drinker mother is less than that born to a non-drinker, the risk of preterm delivery and Low Birth Weight increases in proportion of the severity of anaemia in an anaemic mother. The machine learning models used for training were Multivariate Linear Regression, Multivariate Ridge Regression, K Nearest Neighbours, Decision Trees, Ada-Boost Regressor, and Random Forest Regressor. The machine learning model selected for the final prediction was Random Forest Regressor as the Root Mean Square Error was the least among the models.

3. PROPOSED SYSTEM

The Proposed System is having the following steps to implement Machine learning process.

3.1. Description

In the proposed work, the dataset has to collect from different dataset which includes the major attributes of mother age, gestation weeks, gender and pluralrity of baby from the user. The collected dataset to be analyzed

1.Dataset collection from BigQuery.
2.Analyze the dataset features using Pandas and BigQuery.
-Ensure that dataset have enough examples of each data value, and to verify that the parameter has predictive value.
3.Preprocessing the dataset
-Create training and evaluation dataset.
-Replace missing values with default values.
-Modify plurality field to string.
-Create extra rows to simulate lack of ultrasound

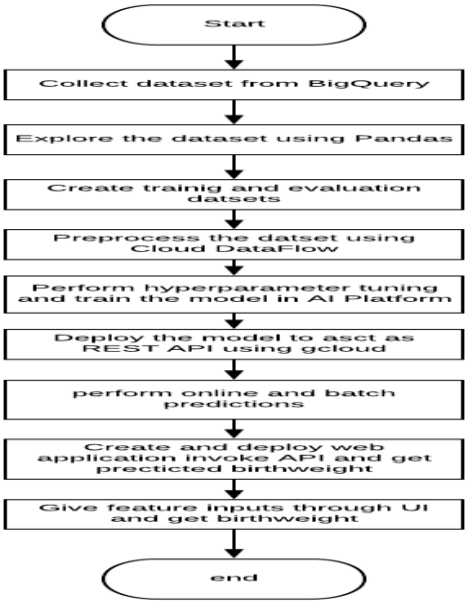

Fig:3 Proposed System

4. Train the model
-Create an input function reading a file using the Dataset API
-Define feature columns
-Create serving input function to be able to serve predictions
-Create metric for hyperparameter tuning
-Create estimator to train and evaluate
-Perform hyperparameter tuning.
-Create python package and submit it to cloud ML engine.

5. Deploy the model as REST API.
-Use model to predict (online and batch prediction)
Develop web application to collect user input and use the API to predict.
-Use model to predict (online and batch prediction)

3.2. Tensorflow

TensorFlow makes it easy to create machine learning models. TensorFlow is an open-source machine learning library for research and production. Estimator is high level API provided by TensorFlow Estimators can train large models on multiple machines in a production environment. TensorFlow provides a collection of pre-made Estimators to implement common ML algorithms. It encapsulates training,evaluation,prediction and export for serving. All Estimators whether pre-made or custom are classes based on the tf.estimator.Estimator class. Pre-made Estimators enable us to work at a much higher conceptual level than the base TensorFlow APIs. Pre-made Estimators create and manage tf.Graph and tf.Session objects. Cloud ML Engine is orthogonal to all APIs in TensorFlow Library. This project uses TensorFlow Version 1.8.0

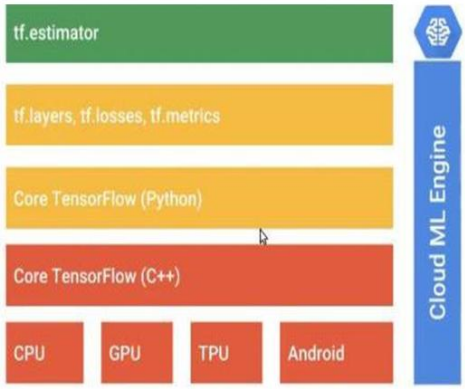

Fig: 4 TensorFlow Architecture

3.2.1. Premade Estimators

Tasks in these estimators include creating input function,defining feature columns,initiating estimator,training,evaluating the model and making predictions from the trained model. The premade estimator we are going to use is DNNLinearCombinedRegressor. It is a regression model since birthweight we are going to predict is a continuous value.Pre-made Estimators are an effective way toquickly create standard models. It also creates checkpoints so that training can be resumed at any time from previous checkpoint.

3.2.2. Checkpoints

Estimators automatically write Checkpoints and event files to disk.Checkpointsare versions of the model created during training. Event filescontain information that TensorBoarduses to create visualizations. The argument named model_dir specifies the directory in which estimators' stores information. If model _dir is not specified in an Estimator's constructor, the Estimator writes checkpoint files to a temporary directory chosen by Python's tempfile.mkdtempfunction. This function picks a secure, temporary directory appropriate for your operating system. By default, it writes a checkpoint every 10 minutes,writes a checkpoint when the train method starts (first iteration) and completes (final iteration) and retains only the 5 most recent checkpoints in the directory. The default schedule can be altered by tf.estimator.RunConfig. Each subsequent call to the Estimator's train, evaluate and predict method builds the model's graphby running the model_fn() and initializes the weights of the new model from the data stored in the most recent checkpoint.

3.2.3. Feature Columns

Feature columns are very rich, enabling you to transform a diverse range of raw data into formats that Estimators can use, allowing easy experimentation. We specify the input to the model through the feature_columns argument. Feature Columns bridge input data with your model. Feature columns are createdusing tf.feature_column module. This module has nine functions. They are Numeric column, Bucketized_column, Categorical identity column, Categorical vocabulary column, Hashed Column, Crossed column,Indicator and embeddingcolumns.The linear_feature_columns argumentacceptsanyfeature column type. The dnn_feature_columns

argument only accepts dense columns. The function categorical_column_with_vocabulary_list maps string to an integer based on an explicit vocabulary list. The function bucketized_column splits column values into different categories based on numerical ranges. The function crossed column combines features into a single feature, better known as feature crosses, enables the model to learn separate weights for each combination of features. The function embedding column never work on features directly, but instead take categorical columns as input. It represents that data as a lower-dimensional, ordinary vector in which each cell can contain any number, not just 0 or 1. By permitting a richer palette of numbers for every cell, an embedding column contains far fewer cells than an indicator column.

3.2.4. Datasets for Estimators

The tf.data module contains a collection of classes thatallows us to easily load data, manipulate it, and pipe it into our model. It enables reading in-memory data from numpy arrays and reading lines from a csv files. The tf.data.TextLineDataset reads the file one line at a time. The Dataset would iterate over the data once, in a fixed order, and only produce a single element at a time. It needs further processing before it can be used for training. Fortunately, the tf.data.Dataset class provides methods to better prepare the data for training. The tf.data.Dataset.shufflemethoduses a fixed- size buffer to shuffle the items as they pass through. In this case the buffer_size is greater than the number of examples in the Dataset, ensuring that the data is completely shuffled. The tf.data.Dataset.repeatmethod restarts the Dataset when it reaches the end. To limit the number of epochs, set the count argument. The tf.data.Dataset.batch method collects a number of examples and stacks them, to create batches. This adds a dimension to their shape. The new dimension is added as the first dimension.Datasets have many methods for manipulating the data while it is being piped to a model. The most heavily-used method is tf.data.Dataset.map, which applies a transformation to each element of the Dataset.The map method takes a map_func argument that describes how each item in the Dataset should be transformed.

3.3. Wide and Deep Neural Network

Generalized linear models with nonlinear feature transformations are widely used for large-scale regression and classification problems with sparse inputs. Memorization of feature interactions through a wide set of cross-product feature transformations are effective and interpretable, while generalization requires more feature engineering effort. With fewer features engineering, deep neural networks can generalize better to unseen feature combinations through low-dimensional dense embeddings learned for the sparse features. However, deep neural networks with embedding can over-generalize and recommend less relevant items when the user-item interactions are sparse and high-rank. Wide & Deep learning jointly trained wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. Memorization can be loosely defined as learning the frequent co-occurrence of items or features and exploiting the correlation available in the historical data.Generalization, on the other hand, is based on transitivity of correlation and explores new feature combinations thathave never or rarely occurred in the past. Compared with memorization, generalization tends to improve the diversity of the system. Embedding-based models, such as factorization machines or deep neural networks, can generalize to previously unseen query-item feature pairs by learning a low-dimensional dense embedding vector for each query and item feature, with less burden of feature engineering. On the other hand, linear models with cross-product feature transformations can memorize these exception rules" with much fewer

parameters. The wide component and deep component are combined using a weighted sum of their output log odds as the pre diction, which is then fed to one common logistic loss function for joint training. Note that there is a distinction between joint training and ensemble. In an ensemble, individual models are trained separately without knowing each other, and their predictions are combined only at inference time but not at training time. In contrast, joint training optimizes all parameters simultaneously by taking both the wide and deep part as well as the weights of their sum into account at training time.

3.4. Dataset

This Project uses publicdata.samples.natality dataset available in BigQuery.The datasetdescribes all United States births registered in the 50 States, the District of Columbia, and New York City from 1969 .It has 137,826,763 rows.The table size is of 21.94 GB.The data location of table is US. It has 31 attributes. But we are interested in 7 main attributes .They are weight in pounds, gestation weeks, plurality, baby's gender, month and year.

Table:1 Schema of Natality table

Field Name	**Type**	**Description**
source_year	INTEGER	Four-digit year of the birth.
year	**INTEGER**	**Four-digit year of the birth.**
month	**INTEGER**	**Month index of the date of birth**
day	INTEGER	Day of birth, starting from 1.
wday	INTEGER	Day of the week from 1 to 7
state	STRING	The two character postal code for the state.
is_male	**BOOLEAN**	**TRUE if the child is male, FALSE if female.**
child_race	INTEGER	The race of the child.
weight_pounds	**FLOAT**	**Weight of the child, in pounds.**
plurality	**INTEGER**	**How many children were born as a result of this pregnancy.**
apgar_1min	INTEGER	Apgar scores measure the health of a new born child on a scale from 0-10.
apgar_5min	INTEGER	Apgar scores measure the health of a newborn child on a scale from 0-10. V
mother_residence_state	STRING	The two-letter postal code of the mother's state of residence when the child was born
mother_race	INTEGER	Race of the mother. Same values as child race.
mother_age	**INTEGER**	**Reported age of the mother when giving birth.**

gestation_weeks	**INTEGER**	**The number of weeks of the pregnancy.**
lmp	STRING	Date of the last menstrual period in the format MMDDYYYY.
mother_married	BOOLEAN	True if the mother was married when she gave birth.
mother_birth_state	STRING	The two-letter postal code of the mother's birth state.
cigarette_use	BOOLEAN	True if the mother smoked cigarettes. Available starting 2003.
cigarettes_per_day	INTEGER	Number of cigarettes smoked by the mother per day
alcohol_use	BOOLEAN	True if the mother used alcohol. Available starting 1989.
drinks_per_week	INTEGER	Number of drinks per week consumed by the mother
weight_gain_pounds	INTEGER	Number of pounds gained by the mother during pregnancy.

Note: Features of interest are highlighted.

The data collected after 2000 is used in this project. Month and year fields are concatenated and the hash is calculated to split the dataset into training and evaluation datasets. FARM_FINGERPRINT is used to find the hash value. The function computes the fingerprint of the STRING or BYTES input using the Fingerprint64 function from the open-source FarmHash library. The output of this function for a particular input will never change. The return type of this function is INT64.

3.5. Process Description

3.5.1.Exploring the Dataset

To train the model, we must explore the dataset, understand its structure, and examine relationships within the data. We then isolate and construct relevant features within the data. A feature is a piece of information that impacts the predictions our model will make. Features can be fields of data in our source dataset, or they can be formed using one or more of the original fields. Identifying the relevant features for our model is called featureengineering. It is done to ensure that dataset have enough examples of each data value, and to verify that the parameter has predictive value. It is also checked whether you have enough for each input value. Otherwise, the model prediction against input values that don’t have enough data may not be reliable.BigQuery python package is imported. Pandas is used to explore the dataset. Dataset is retrieved from BigQuery and stored in pandas Data frame. Then the data is visualised using plot function provided by Pandas. Data collected above 2000 is used for training.The number of records and the average weight for each value of the separate features is found.

From the exploration, it is interpreted the following:

-Male babies are heavier on average than female babies
-Teenaged and older moms tend to have lower-weight babies
-Twins, triplets, etc. are lower weight than single births.

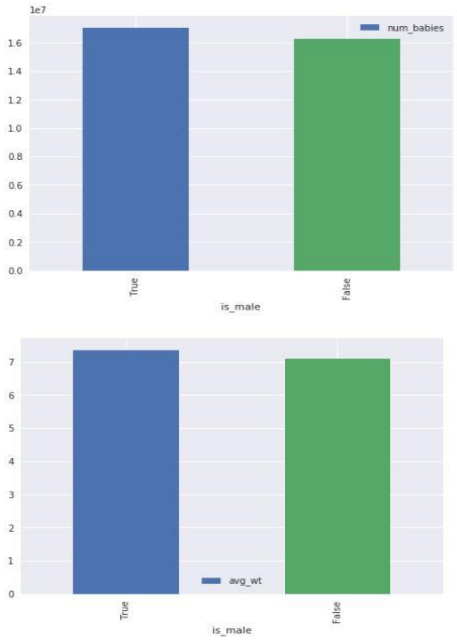

Fig: 5 Exploring is_male feature

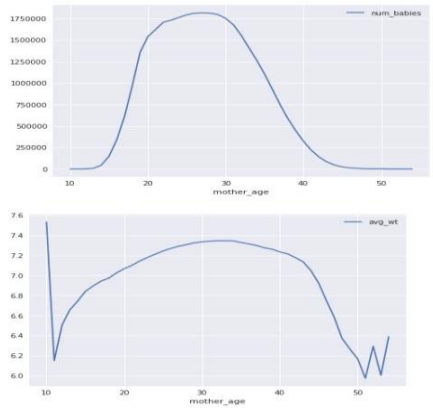

Fig:6 Exploring mother_age feature

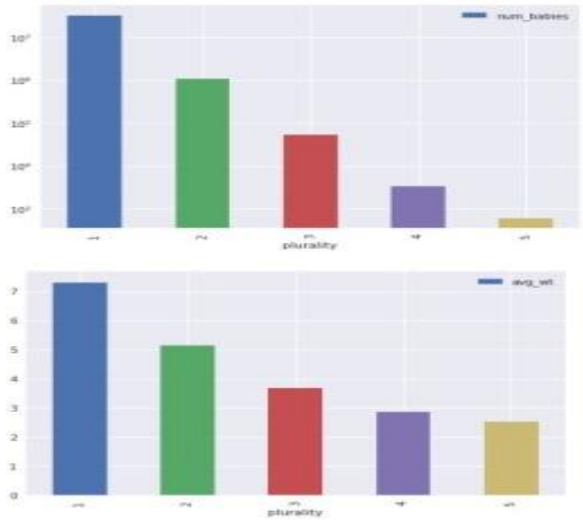

Fig:7 Exploring plurality feature

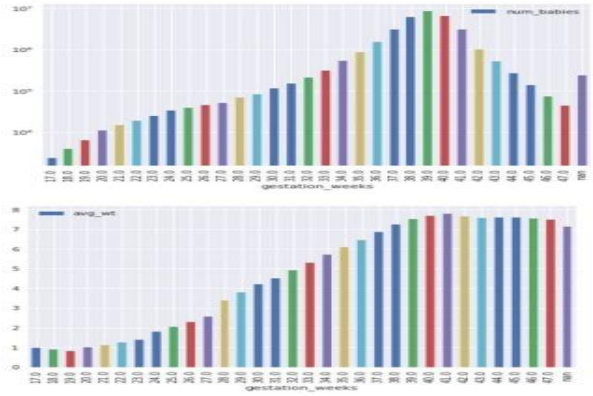

Fig:8 Exploring gestation weeks feature

3.5.2. Preprocessing the Dataset

Pre-processing is done to transform data into a format suitable for training. In this step, dataset is split into training and evaluation datasets. They are created using hash function and modulo function. The hash of the year-month is used so that twins born on the same day won't end up in different cuts of the data. The hash is calculated using FARM_FINGERPRINT function. The quarter of data is used for evaluation and remaining are used for training. Hence the remainder of dividing the hash by is used to define the two datasets. If the remainder is less than 3, that record belongs to training set and if it is equal to 3, that record belongs to evaluation set. This technique ensures that we get a random sampling of the source data in each dataset and reduces the risk of accidentally skewing the evaluation set. . It is important to guarantee that the evaluation set represents the general characteristics of the data so that you can evaluate the generalization performance of the trained model with it.The columns are pulled out of BigQuery and stored in a CSV file within a Cloud Storage bucket. Cloud Dataflow is used to generate synthetic data to make the model more robust to partial or unknown input values.

In the dataset, every row in the dataset contains the baby's gender, because this is known after the baby is born. However, we are building a model to predict the weight before the baby is born. We know the sex of the baby only if an ultrasound was performed during the pregnancy. If no ultrasound was performed, the doctor enters the baby's gender as "Unknown".So we generate artificial data by writing each historical data point twice, once with the original value for the is_male column and again after replacing the is_male column value by Unknown.

Also, it is difficult to count the number of babies without an ultrasound, so while doctors can tell whether there is one baby or multiple babies, they can't differentiate between twins and triplets.We replace the plurality numbers with string values when writing out the data to simulate the absence of an ultrasound.All these code are written with Apache Beam SDK.Apache Beam SDK is programming model for both batch and streaming use cases that implements data processing jobs that run on any execution engine and executes pipelines on multiple execution environments. The job is submitted to cloud dataflow using DataflowRunner which takes about 30 minutes to finish. Job details are showed in Cloud Dataflow page in Google cloud console.

Fig: 9 Cloud Dataflow data processing pipeline

3.5.3. Training the Model

The model is trained in cloud environment using AI Platform. The model code is created as python package and submitted to AI Platform using gcloud tool.The numerical features are mother_age and gestation_weeks.The categorical features are is_male and gestation_weeks.One hot encoding is applied to categorical columns.Wide model woks well on categorical features.Hence numeric features are Bucketized and passed to wide model.On the other hand deep model works well on numeric features. Wide features are crossed, embedded and passed as additional input to deep model. An input_fn is created to return batch of examples for training for each invocation. It identifies files that match the filename pattern and shuffles them before retrieving examples. After reading a batch of rows from a CSV file, tf.decode_csv converts column values into a list of TensorFlow constant objects. The DEFAULTS list is used to identify the value types and complement empty cells. The input_fnfunction returns the dictionary of features and the corresponding label values.Metrics for hyperparameter tuning is added with the parameter to be optimized as root mean squared error. Then serving input function is created to serve predictions using user inputs. Estimator object is initiated with

DNNLinearCombinedRegressor as model specifying model directory,features and hidden units. Training and evaluation specifications are specified. Training and evaluation of the model is done by calling tf.estimator.train_and_evaluate function. All these details are specified in model.py file. The python package contains 3 files-__init__.py to inform that it is a python package, model.py and trainer.py for handling command line arguments which reads value for parameters and sets them to the appropriate argument in model.py file. The arguments are bucket name,output directory,batch size,train examples,evaluation steps,file pattern,nembeds and neural network size.The training job is submitted to AI Platform using gcloud specifying package path, command line arguments etc.Once training is started,it can be visualized using tensor board tool.

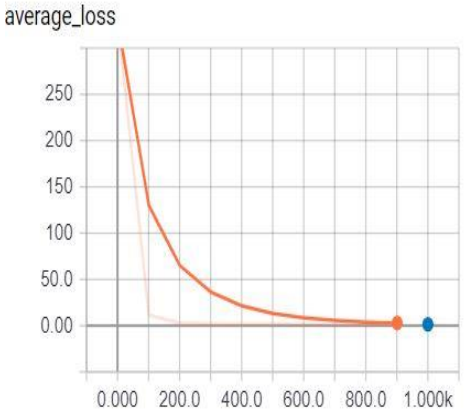

Fig:10 average loss during training

3.5.4. Deploying Model

The trained model is stored in exporter directory. This directory is given as option for gcloud command. The gcloud command is used for creating model and then version.The models are listed in AI Platform under the models tab. After that online and batch predictions are performed. For online prediction, json object is created with the features and sent with the request to the API.For batch predictions,an input file is created and outputpredictions are stored in cloud storage. Authentication is required for accessing the API.For that purpose access token is passed with the headers. Access token is generated using Google Credentials module.

3.5.5. Creating a Web Application

GoogleAppEngine is used to deploy a web application. The file deploy.sh contains code for creating and deploying the app. The app is deployed in asia-south1 region. The main.py is a Python script that runs on App Engine. It provides an API service that returns a prediction for a baby's weight. To get this predicted value, it uses the prediction API service deployed on the Managed ML Service.It sends a request to the prediction API service hosted on Cloud ML Engine converting plurality and gender field to string.It also checks whether all the fields are set.It uses Google Credentials for authorization.The templates/form.html is an HTML file containing JavaScript code that renders the input form shown below. It sends a REST API request to the backend application that runs on App Engine and then displays the result. The application is autoscalable and provides high performance.It supports several queries in a second.

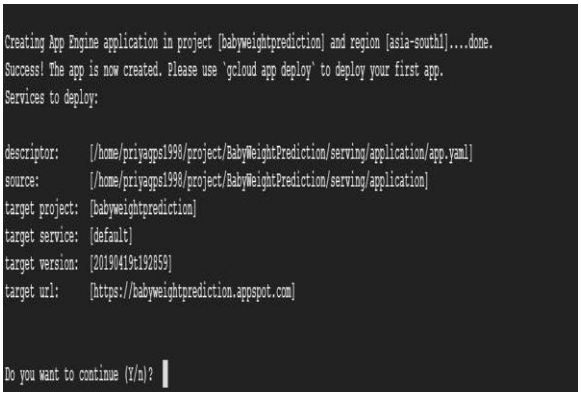

Fig:11 App Details

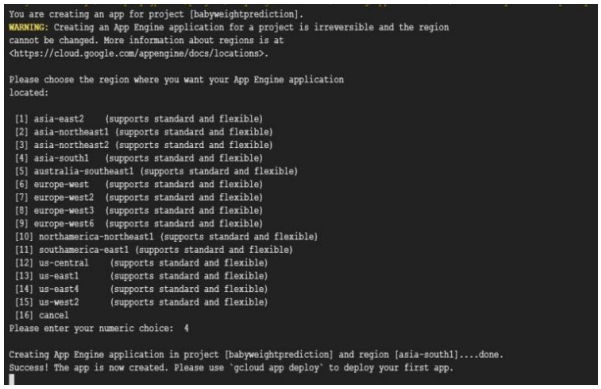

Fig:12 Regions available for deploying application

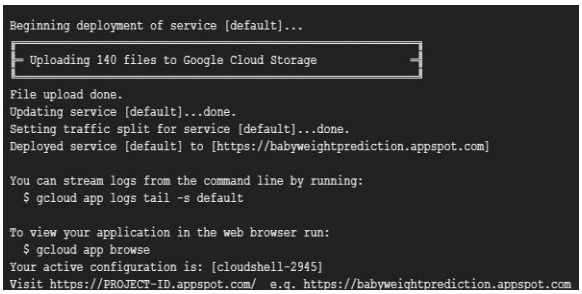

Fig:13 Deployment of application

3.6. Tools, Libraries and Apis Used

Bigquery

BigQuery is Google's serverless, highly scalable, enterprise data warehouse designed to make all our data analysts productive at an unmatched price-performance. There is no infrastructure to manage, we can focus on analysing data to find meaningful insights using familiar SQL without the need for a database administrator.We can analyse all our data by creating a logical data

warehouse over managed, columnar storage, as well as data from object storage and spreadsheets. It enables us to build and operationalize machine learning solutions with simple SQL. We can easily and securely share insights as datasets, queries, spreadsheets, and reports. BigQuery allows organizations to capture and analyze data in real time using its powerful streaming ingestion capability so that your insights are always current, and it's free for up to 1 TB of data analyzed each month and 10 GB of data stored. It enables us to save query, validate query, save view, schedule query, export table and get summary of query. It shows job history and explore in data studio.

BigQuery supports a standard SQL dialect which is ANSI: 2011 compliant, reducing the need for code rewrite and allowing us to take advantage of advanced SQL features. BigQuery provides free ODBC and JDBC drivers to ensure our current applications can interact with Big Query's powerful engine.BigQuery provides rich monitoring, logging, and alerting throughStackdriver Audit Logs. BigQuery resources can be monitored at a glance, and BigQuery can serve as a repository for logs from any application or service using Stackdriver Logging.

Public Dataset

A public dataset is any dataset that is stored in BigQuery and made available to the general public through the Google Cloud Public Dataset Program. The public datasets are datasets that BigQuery hosts for you to access and integrate into our applications. Google pays for the storage of these datasets and provides public access to the data via a project. Public datasets are available for you to analyze using either legacy SQL or standard SQL queries. You can access BigQuery public data sets by using the BigQuery web UI in the GCP Console, the classic BigQuery web UI, the command-line tool, or by making calls to the BigQuery REST API using a variety of client libraries such as Java, .NET, or Python.

In addition to the public datasets, BigQuery provides a limited number of sample tables that you can query. These tables are contained in the bigquery-public-data :samples dataset. It includes the following tables gsod,github_nested,githubtimeline,natality,Shakespeare,trigrams,Wikipedia.This project uses natality dataset for training the model.

Cloud Datalab

Cloud Datalab is a powerful interactive tool created to explore, analyze, transform and visualize data and build machine learning models on Google Cloud Platform. It runs on Google Compute Engine and connects to multiple cloud services easily so we can focus on our data science tasks. Cloud Datalab is built on Jupyter (formerly IPython), which boasts a thriving ecosystem of modules and a robust knowledge base. Cloud Datalab enables analysis of our data on Google BigQuery, Cloud Machine Learning Engine, Google Compute Engine, and Google Cloud Storage using Python, SQL, and JavaScript (for BigQuery user-defined functions). Whether we are analyzing megabytes or terabytes, Cloud Datalab has you covered. Query terabytes of data in BigQuery, run local analysis on sampled data and run training jobs on terabytes of data in Cloud Machine Learning Engine seamlessly.This project uses Cloud Datalab in asia-south1-a region to create notebooks and execute code.

Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. This project uses pandas to explore dataset.It supports three datastructures- Dataframe,series, and panel. A Data frame is a two-dimensional data structure.It supports columns of different types and can perform arithmetic operations on rows and columns.It helps data visualisation using matplotlib libraries plot() method .

Google Cloud Storage

Google Cloud Storage is a RESTful online file storage web service for storing and accessing data on Google Cloud Platform infrastructure. The service combines the performance and scalability of Google's cloud with advanced security and sharing capabilities. It is an Infrastructure as a Service, comparable to Amazon S3 online storage service. Contrary to Google Drive and according to different service specifications, Google Cloud Storage appears to be more suitable for enterprises. Google Storage (GS) stores objects that are organized into buckets identified within each bucket by a unique, user-assigned key. All requests are authorized using an access control list associated with each bucket and object. Bucket names and keys are chosen so that objects are addressable using HTTP URLs. Google Storage offers four storage classes, identical in throughput, latency and durability. The four classes, Multi-Regional Storage, Regional Storage, Near line Storage, and Cold line Storage, differ in their pricing, minimum storage durations, and availability. This project stores pre-processed dataset, trained model, hyperparameter tuning details, application in a Multi-regional bucket.

Compute Engine Api

It creates and runs virtual machines on Google Cloud Platform. Compute Engine’s tooling and workflow support enable scaling from single instances to global, load-balanced cloud computing. Compute Engine's VMs boot quickly, come with persistent disk storage, and deliver consistent performance. Our virtual servers are available in many configurations including predefined sizes or the option to create Custom Machine Types optimized for your specific needs. Flexible pricing and automatic sustained use discounts make Compute Engine the leader in price/performance. It is a billable component. It must be enabled to use Datalab. It enables us to resize our clusters, create machine images, virtualize our network, use Pre-emptible for batch workloads and create Custom Machine Types to optimize for our specific needs.

Cloud Dataflow

Cloud Dataflow is a fully-managed service for transforming and enriching data in stream (real time) and batch (historical) modes with equal reliability and expressiveness -- no more complex workarounds or compromises needed. And with its serverless approach to resource provisioning and management, you have access to virtually limitless capacity to solve your biggest data processing challenges, while paying only for what you use. Cloud Dataflow supports fast, simplified pipeline development via expressive SQL, Java, and Python APIs in the Apache Beam SDK, which provides a rich set of windowing and session analysis primitives as well as an ecosystem of source and sink connectors. Plus, Beam’s unique, unified development model lets you reuse more code across streaming and batch pipelines. GCP’s serverless approach removes

operational overhead with performance, scaling, availability, security and compliance handled automatically so users can focus on programming instead of managing server clusters. Integration with Stackdriver, GCP's unified logging and monitoring solution, lets you monitor and troubleshoot your pipelines as they are running. Rich visualization, logging, and advanced alerting help you identify and respond to potential issues. This project uses Dataflow runner to pre-process dataset in cloud Dataflow which stores the processed dataset in bucket.

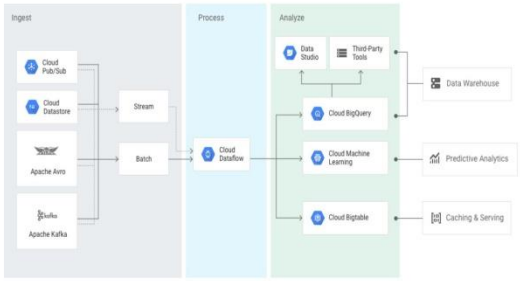

Fig: 14 Data transformation with cloud Dataflow

Cloud Ml Engine

Cloud Machine Learning Engine is a managed service that lets developers and data scientists build and run superior machine learning models in production. Cloud ML Engine offers training and prediction services, which can be used together or individually. We can scale up model training by using the Cloud ML Engine training service in a serverless environment within GCP. Cloud ML Engine supports popular ML frameworks or lets you run our application within a Docker image. It also provides built-in tools to help us understand our models and effectively explain them to business users. Cloud ML Engine automatically sets up an environment for XGBoost and TensorFlow to run on multiple machines, so you can get the speed you need by adding multiple GPUs to our training job or splitting it across multiple VMs. It helps us achieve better results faster by automatically tuning deep learning hyperparameters with HyperTune. HyperTune saves many hours of tedious and error-prone work.Once we have a trained model, Cloud ML Engine offers two types of predictions to apply what the computer learned to new examples.

Online Prediction deploys ML models with serverless, fully managed hosting that responds in real time with high availability. Our global prediction platform automatically scales to adjust to any throughput. It provides a secure web endpoint to integrate ML into your applications. Batch Prediction offers cost-effective inference with unparalleled throughput for asynchronous applications. It scales to perform inference on TBs of production data.

Cloud ML Engine has deep integration with our managed notebook service and our data services for machine learning: Cloud Dataflow for feature processing, BigQuery for dashboard support and analysis, and Cloud Storage for data storage. It is used for training and deploying model.

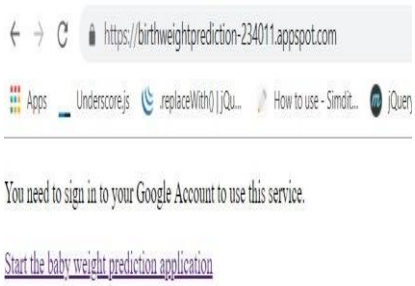

Fig:15 Index page

The users are redirected to the authorization server where the user provide their Google account credentials and authorized.

Fig: 14 Main Page

User needs to set all the fields otherwise it asks the user to set all items.

Fig: 15 Baby weight predictor

4. CONCLUSION AND FUTURE WORK

This application predicts the birthweight within a minute. It helps in making prior arrangement of doctors and facilities before baby birth. It is available all time since it is deployed in the cloud. This application is helpful mainly for hospitals, pregnant ladies. It is highly scalable and provides high performance and maintainability. This application currently depends on the four major factors mother's age, gestation weeks, plurality and gender of baby. In future, features will extended including several other factors like smoking mother, mother's race, country, etc. so that the algorithm will be improved to get the best results. Also, this web application will be extended as mobile application to improve portability and flexibility. It will be improved in such a way that it provides suggestions to prevent low birthweight and helps mother take special care during pregnancy.

REFERENCES

[1] Alfensi Faruk, Endro Setyo Cahyono, Ning Eliyati,IkaArifieni,(2018),"Prediction and Classification of LowBirth Weight DataUsing Machine LearningTechniques" Indonesian Journal of Science &Technology 3(1)18-28.

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah,"Wide & Deep Learning for Recommender Systems" - arXiv: 1606.07792v1 [cs.LG]

[3] Stefan Kuhle, Bryan Maguire, Hongqun Zhang, David Hamilton, Alexander C. Allen, K.S.Joseph, Victoria M. Allen,,"Comparision of logistic regression with machine learning methods for the prediction of fetal growth abnormalities:a retrospective study ", BMC Pregnancy and Childbirth, 18(1):333. doi: 10.1186/s12884-018-1971-2.

[4] Dahlui, M., Azahar, N., Oche, O. C., and Aziz, N. A. "Risk factors for low birth weight in nigeria: evidence from the 2013 Nigeria demographic and health survey. Global Health Action",2016, 9, 28822.

[5] Firdaus, C., Wahyudin, W., & Nugroho, E. P. (2017)."Monitoring System with Two Central Facilities Protocol." Indonesian Journal of Science and Technology, 2(1), 8-25.

[6] https://codelabs.developers.google.com

[7] https://cloud.google.com/solutions/machinelearning/data-preprocessing-for-ml-with-tf-transform-pt2

[8] https://www3.cs.stonybrook.edu/~skiena/591/final_projects/baby_weight/

[9] https://cloud.google.com/mlengine/docs/tensorflow/hyperparameter-tuning-overview