Context Recognition Using Smartphone Sensor Data

Team ContextualHealing : Abhishek, Anant, Brandon, Hank

Introduction

Context Recognition deals with automatic recognition of a user's context using data

Physical Activity

Location

Time

Market Potential

Context Aware Apps

$77 B

Mobile apps market

Healthcare

2 M

Deaths due to physical inactivity

Smart Mobile Ads

$ 6 B

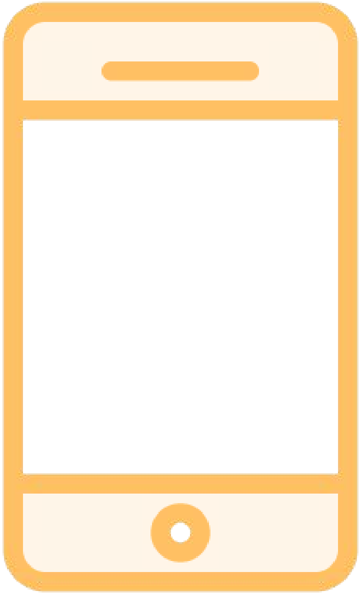
Mobile ads market

Source : http://www.gartner.com/newsroom/id/2654115
Source : https://blog.gyminsight.com/859-most-current-fitness-industry-statistics/

Why Smartphone for Context Recognition ?

79% 0f people ages 18-44 have their

Smartphones with them 22 hours a day



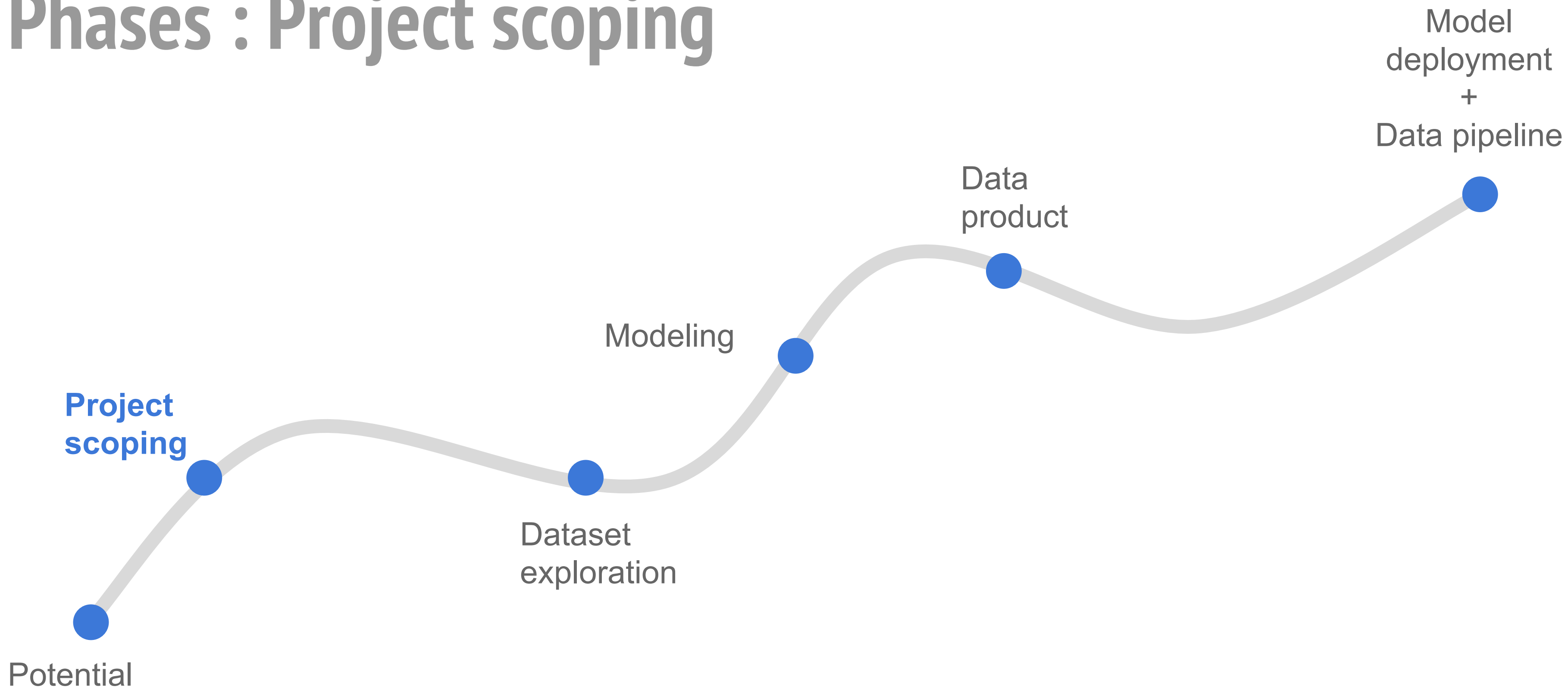
Souce : http://www.adweek.com/socialtimes/smartphones/480485

Goal

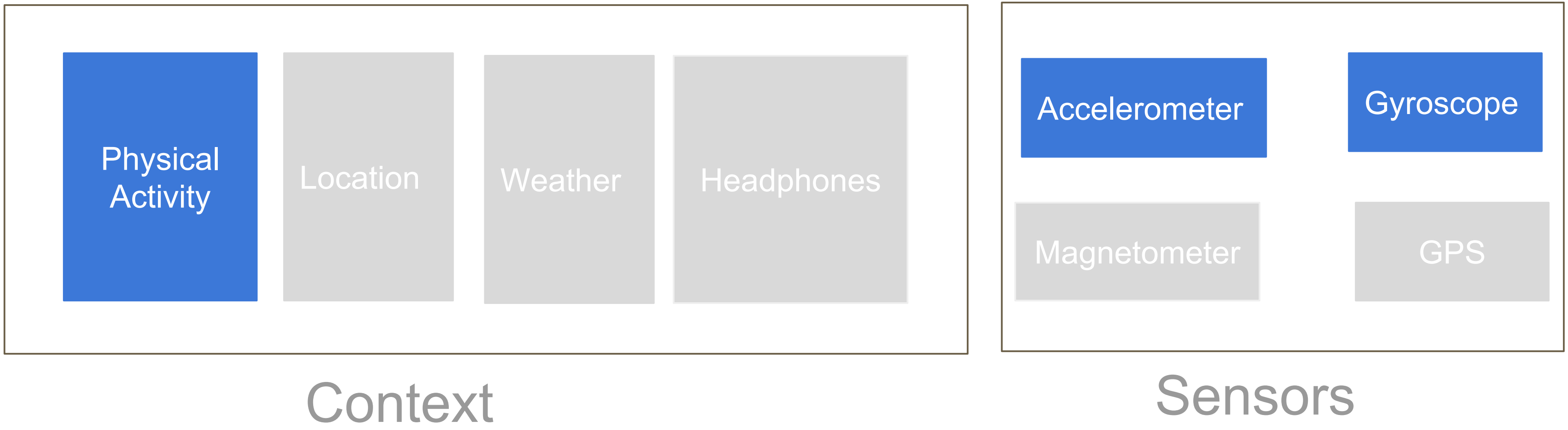
Build a context recognition service running on a predictive modeling framework developed using publicly available datasets.

Create an android application utilizing the modeling backend for tracking user activity.

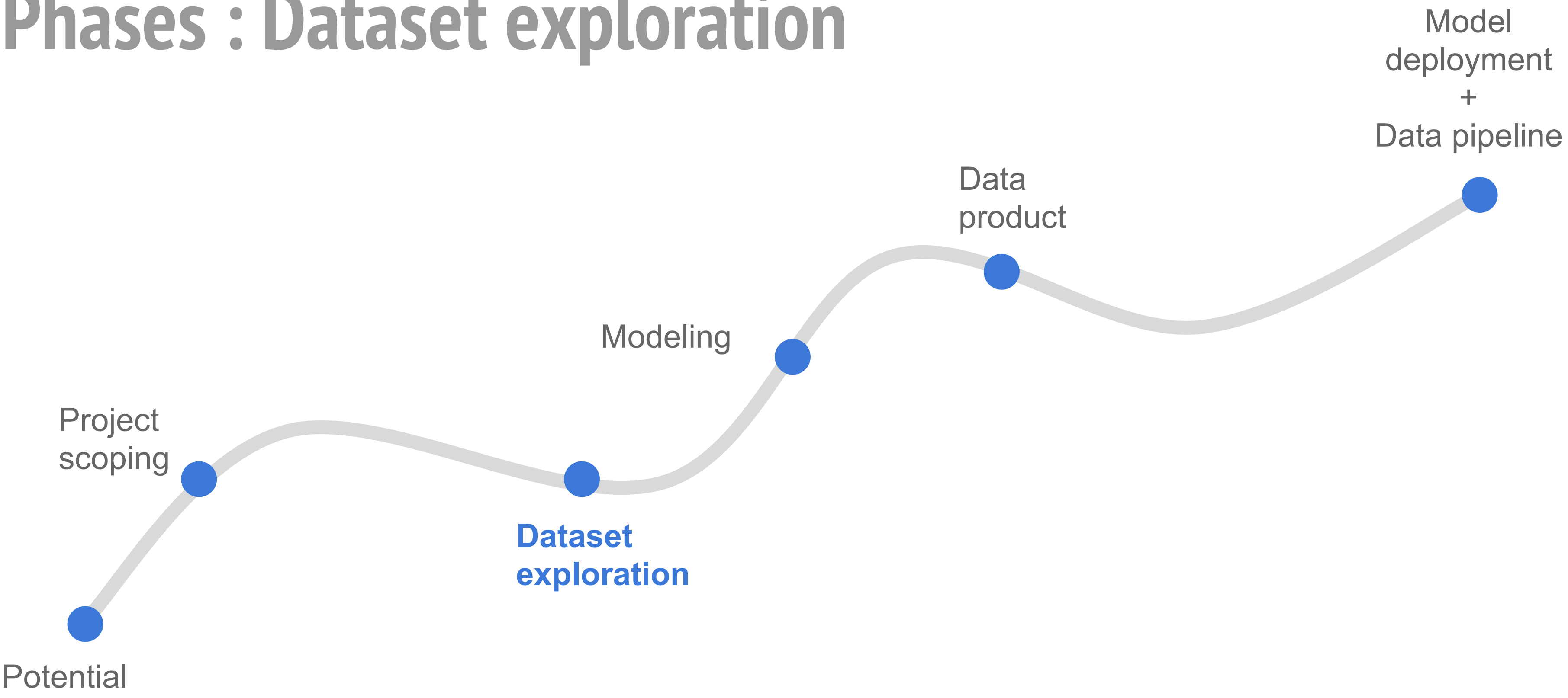
Phases : Project scoping



Project Scope



Phases : Dataset exploration



Datasets

Actitracker

1.1 M

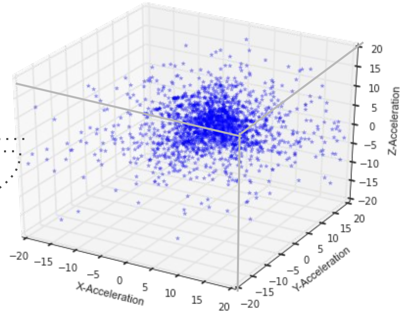
Rows of raw accelerometer data at 50ms

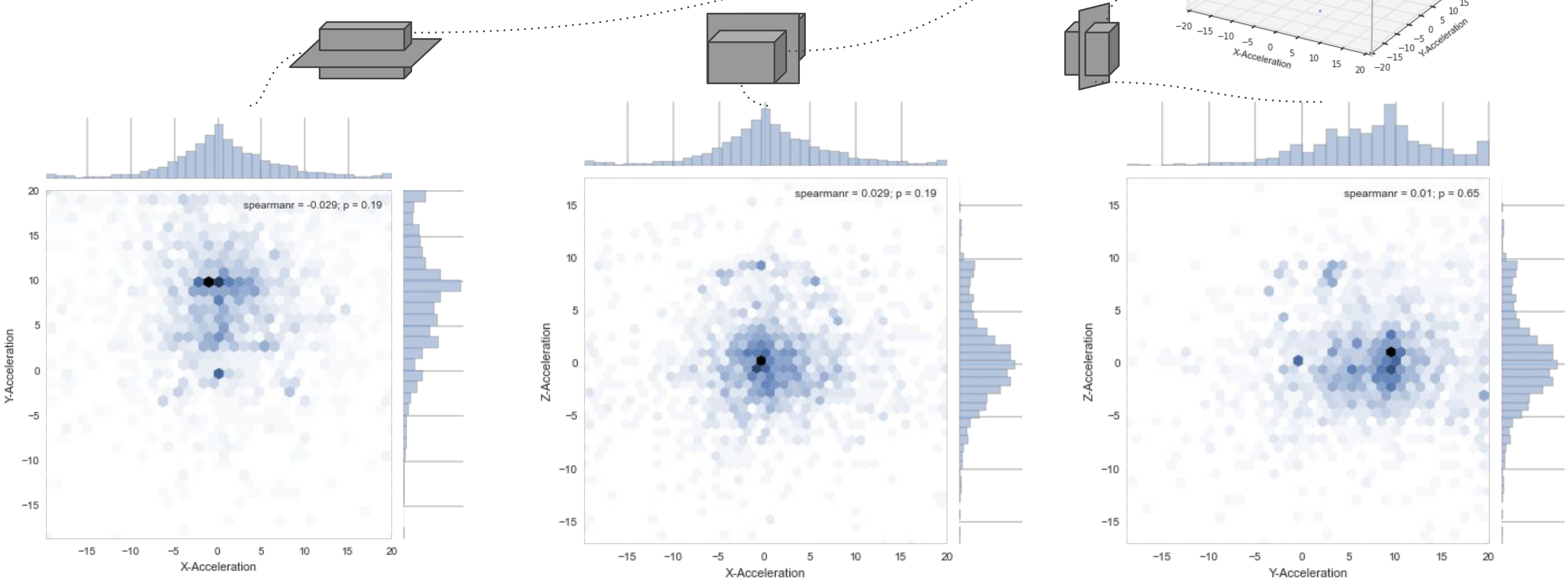
UCI

7 K

Rows of aggregated accelerometer & gyroscope data

Exploratory Analysis



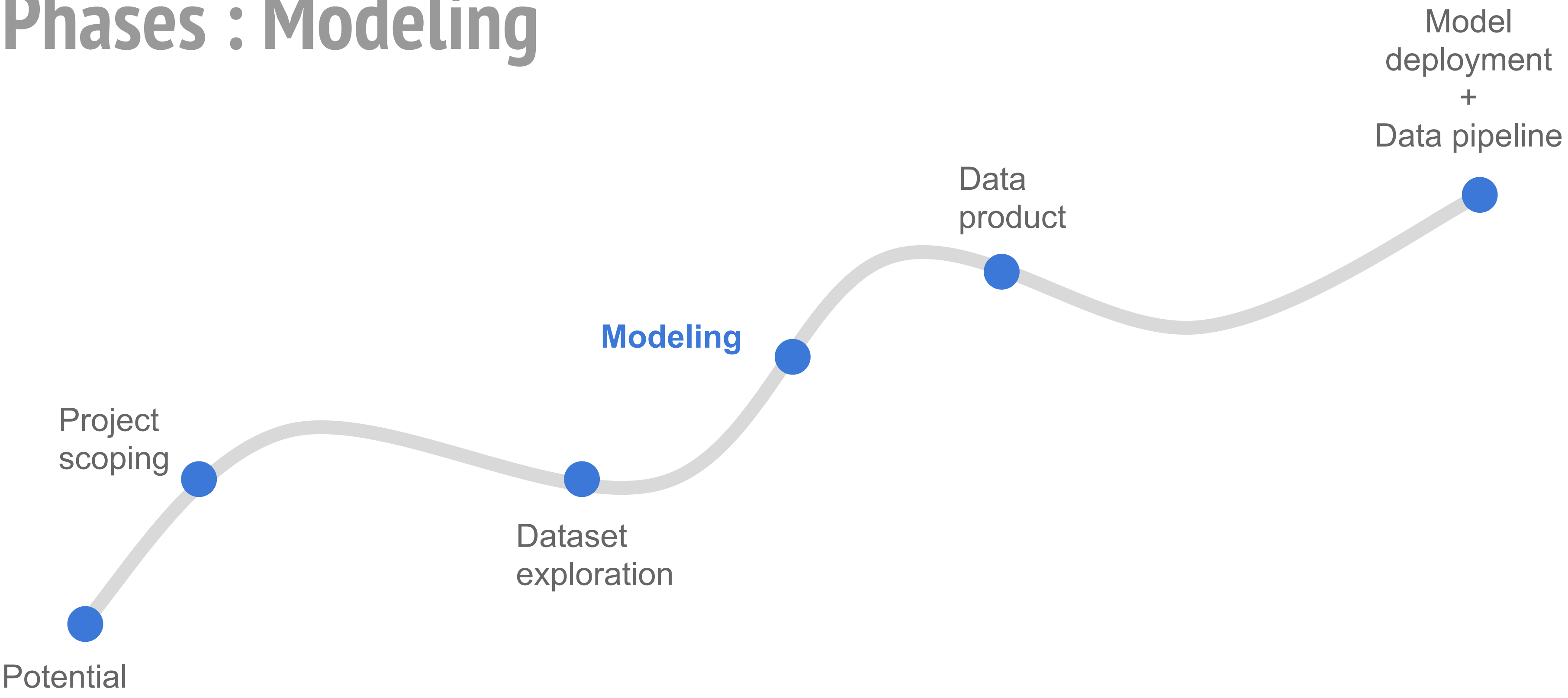


Note: For charts, data is sampled with N=2000

Feature Engineering

- Raw data grouped into 5-second ‘sessions’ (100 raw sensor readings per session)
- For each session, 58 total features:
  - Descriptive statistical features [24]:
    - Mean, median, standard deviation
    - IQR, min, max
    - Kurtosis, skewness
  - Deciles [27]
    - 10th, 20th, 30th, 40th, 50th, 60th, 70th, 80th, 90th percentiles for each axis & session
  - Average Absolute Difference [3]:
    - Average absolute difference between the value of each of the 100 readings within the session and the mean value over those 100 readings
  - Average Resultant Acceleration [1]:
    - Average of the square roots of the sum of the readings of each axis squared $\sqrt{x^2 + y^2 + z^2}$
  - Time Between Peaks [3]
    - Time between maximum and minimum sensor readings in each session

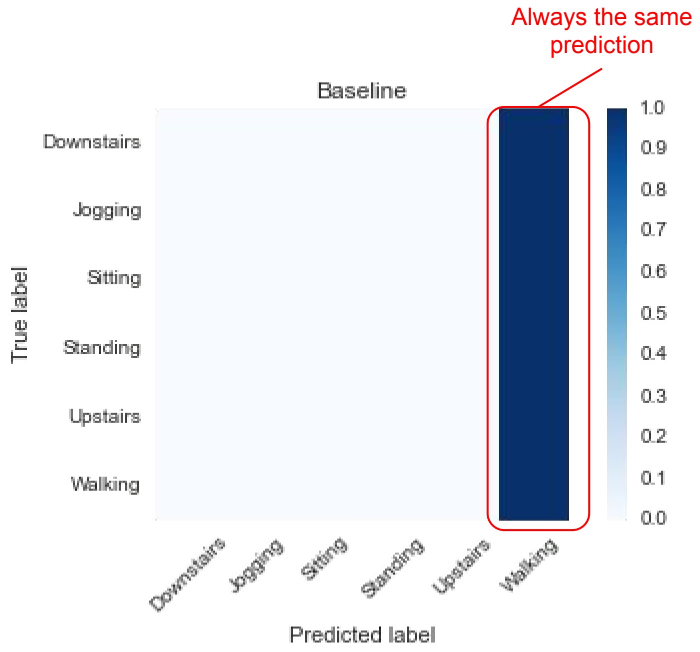
Phases : Modeling



Models : Baseline Model

Baseline

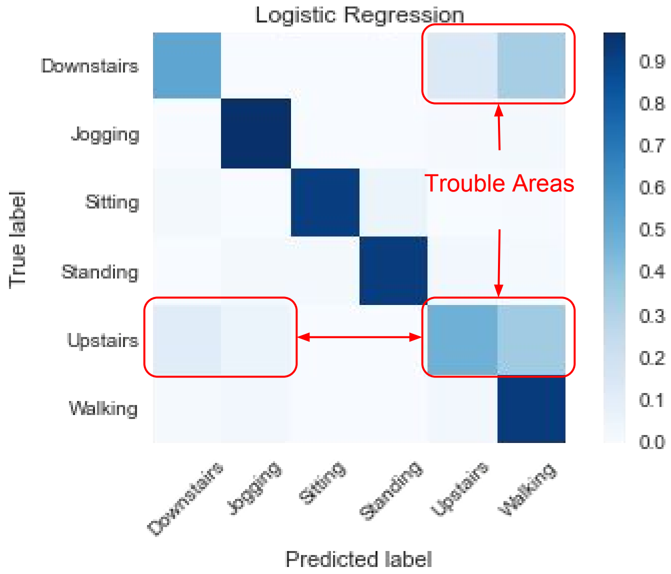
Accuracy : 39.4%
Log-Loss : 1.478



Models : Logistic Regression

Logistic Regression

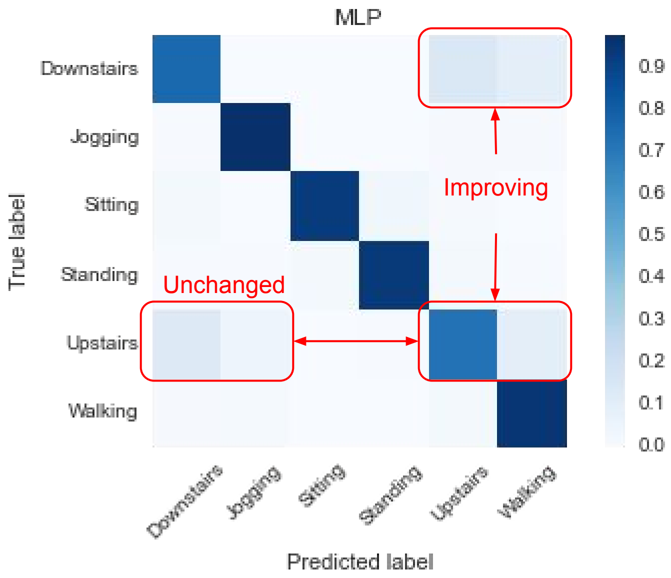
Accuracy : 85.3 %
Log-Loss : 0.509



Models : Multilayer Perceptron

Multilayer Perceptron

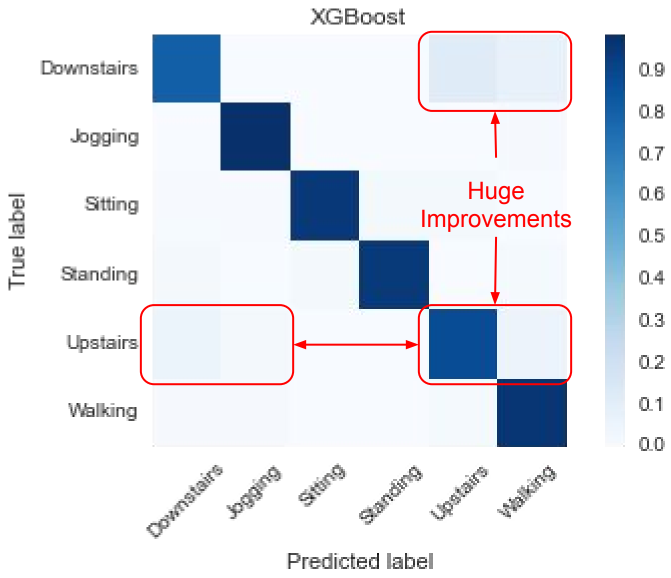
Accuracy : 91.7 %
Log-Loss : 0.293



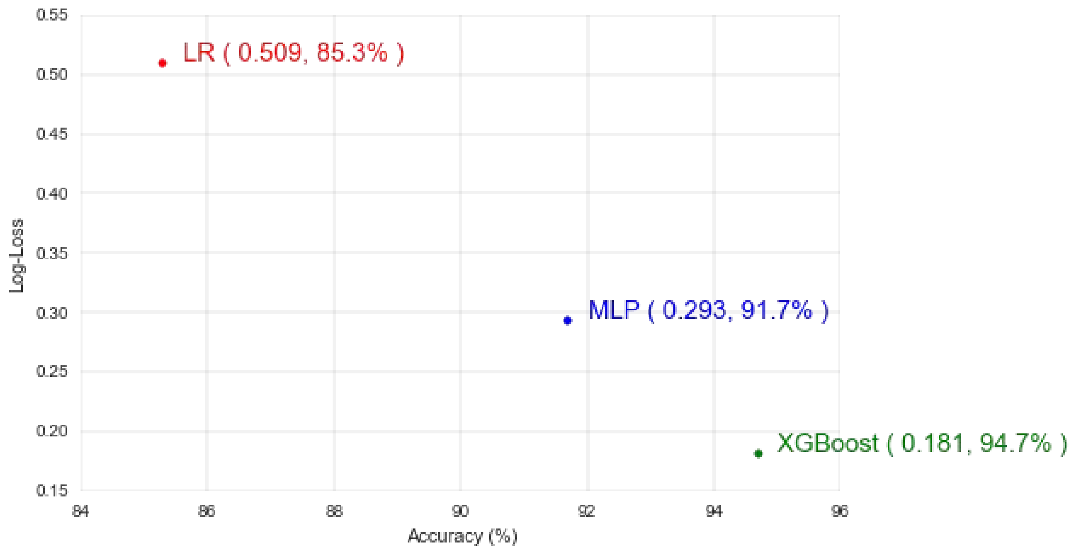
Models : XGBoost

Boosted Trees

Accuracy : 94.7 %
Log-Loss : 0.181



Model Comparisons

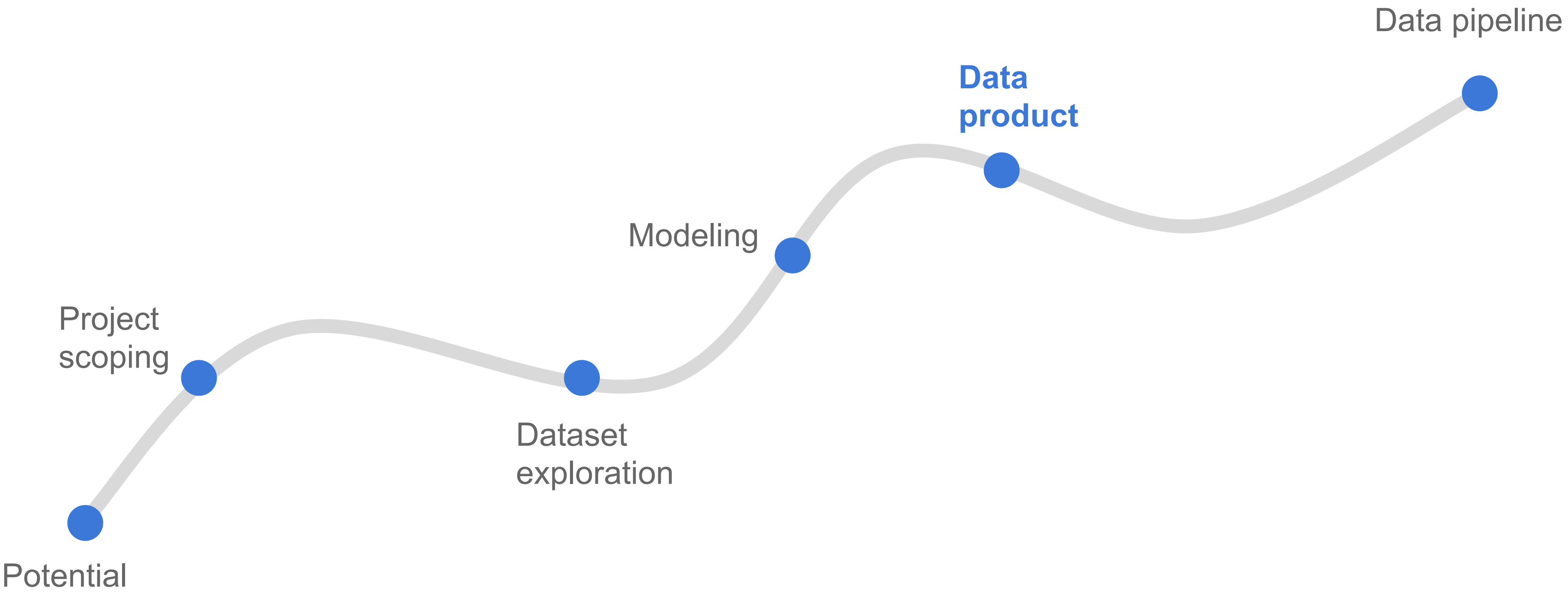


Overall Model Performances

Error Rate Comparisons

	LR	MLP	XGBoost	Δ
Upstairs	52.5%	27.2%	12.2%	**-77%**
Downstairs	48.0%	23.6%	19.3%	**-60%**
Walking	7.1%	4.7%	3.6%	-49%
Jogging	2.2%	2.5%	1.2%	-45%
Sitting	8.3%	6.7%	4.7%	-43%
Standing	7.8%	5.8%	5.2%	-33%

Phases : Data Product



Data Product -1 : Android App

GoalTick

The smart personal assistant to tick all your goals

Why Another Fitness/Activity Tracking App ?

Costly wearables

Require internet connection

Moves

Nike +

Google Fit **

** Just have started supporting offline tracking for limited activities but sends out data to the server

GoalTick

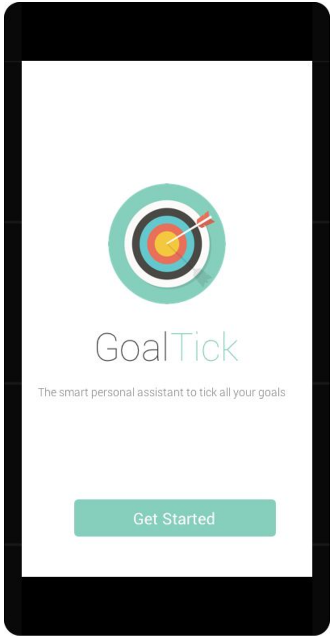
- Set your activity goals
- App will track the activity
- Prompt user in between
- Provide summary results
- UI developed using Android Material Design
- Supports both local and server mode for predictions

Machine Learning
On
Device

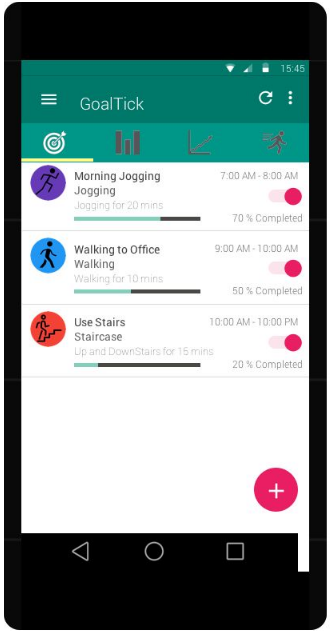
- Tensorflow C++ libraries
- Prediction call written in C++
- Uses JNI (Java Native Interface) to bridge the gap
- Model saved as Protobuf file

UI Prototype

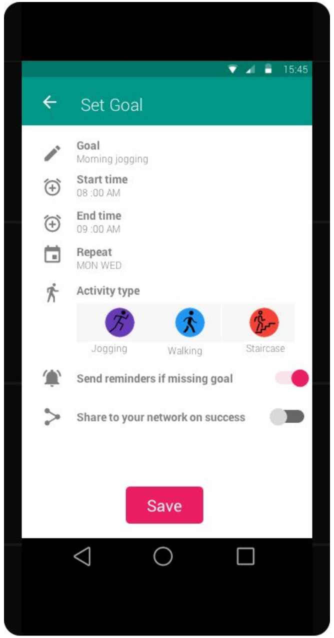
Link to interactive prototype



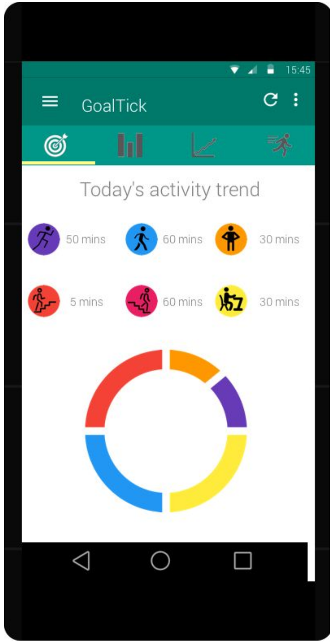
Splash screen



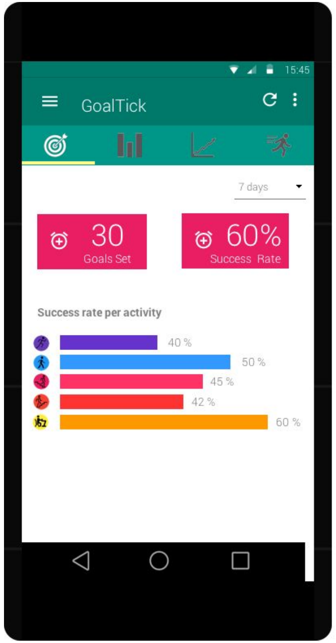
Active goals



Create goal



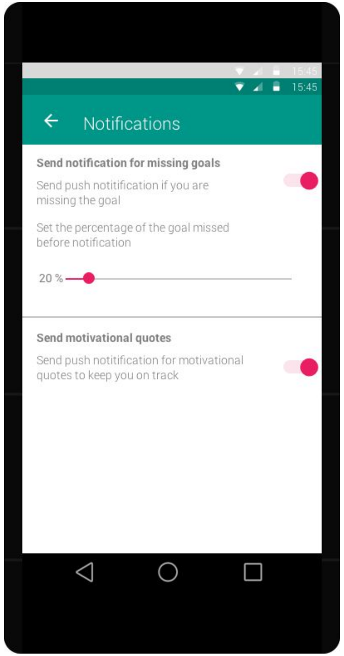
Activity trend



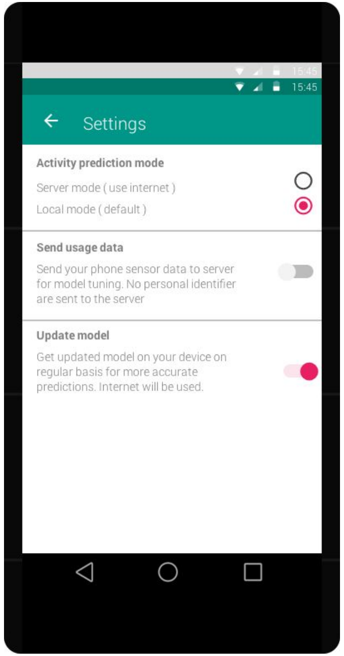
Report

UI Prototype

[Link to interactive prototype]



Notifications



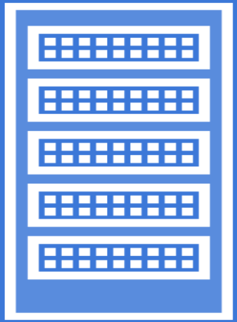
Settings

Data Product : GoalTick

Link to interactive prototype

GoalTick

GET IT ON
Google Play

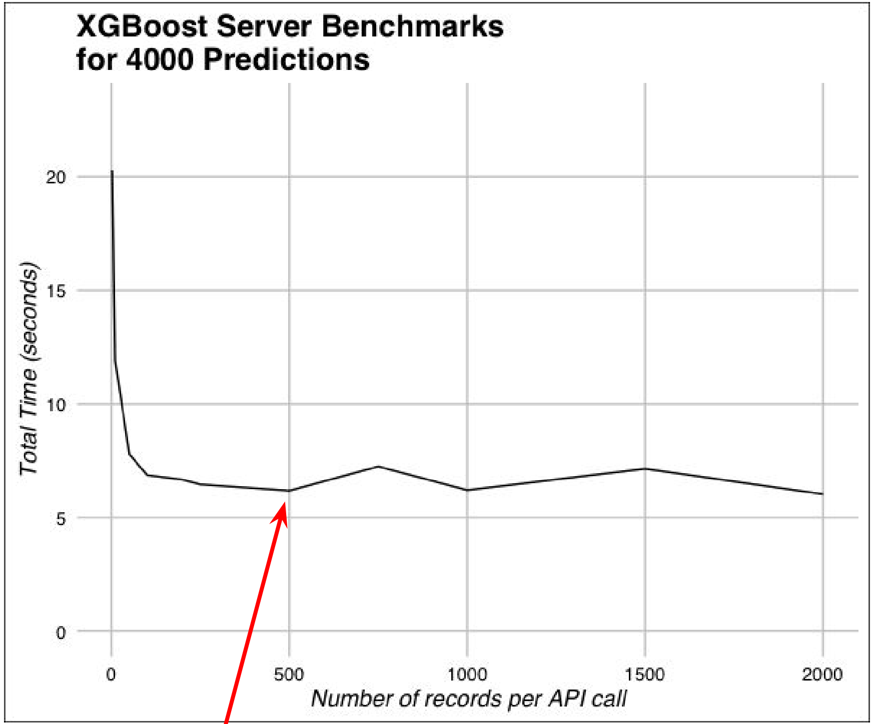


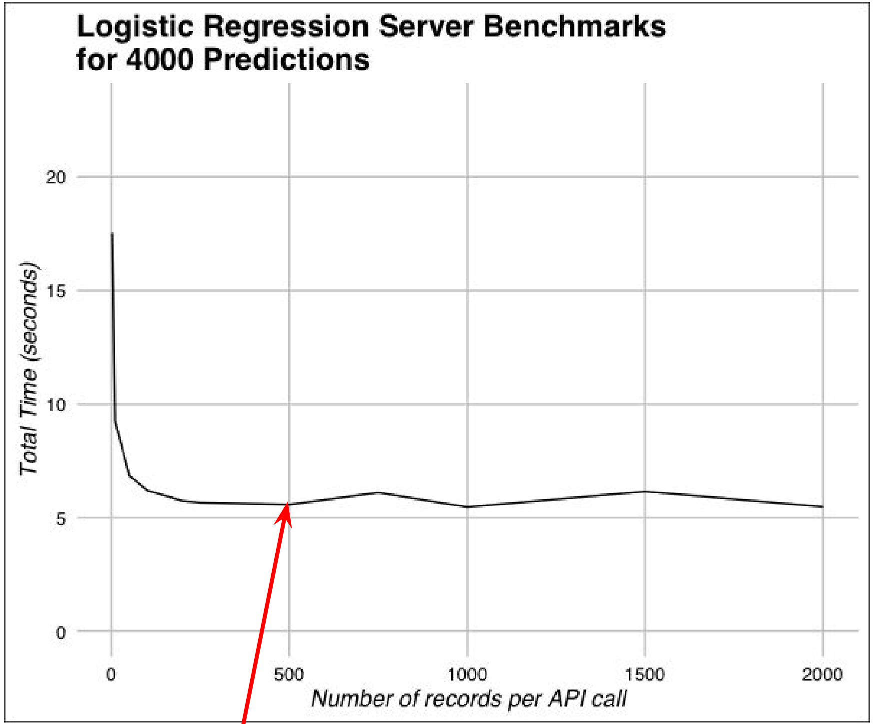
Model Deployment :
REST API

Data Product -2 : Activity Prediction Service

- Flask-based REST API using python
- Endpoint exposed for single or multiple predictions
- For enterprise customers (for bulk predictions)

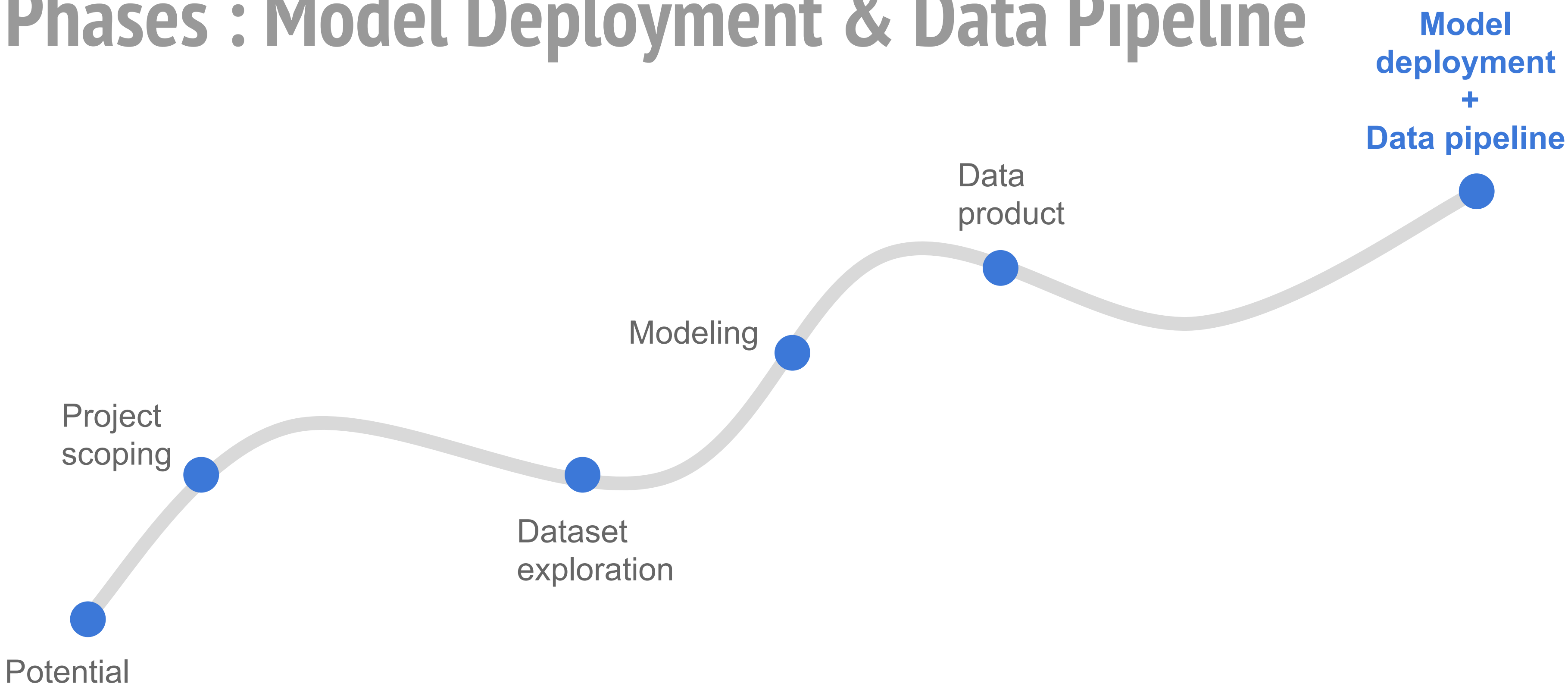
REST API Performance for 5.5 Hours of Data



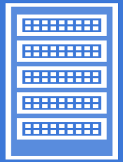


Total prediction time stops improving at around 500 predictions per API request (Approx. 40 minutes)

Phases : Model Deployment & Data Pipeline



Model Deployment: Modes



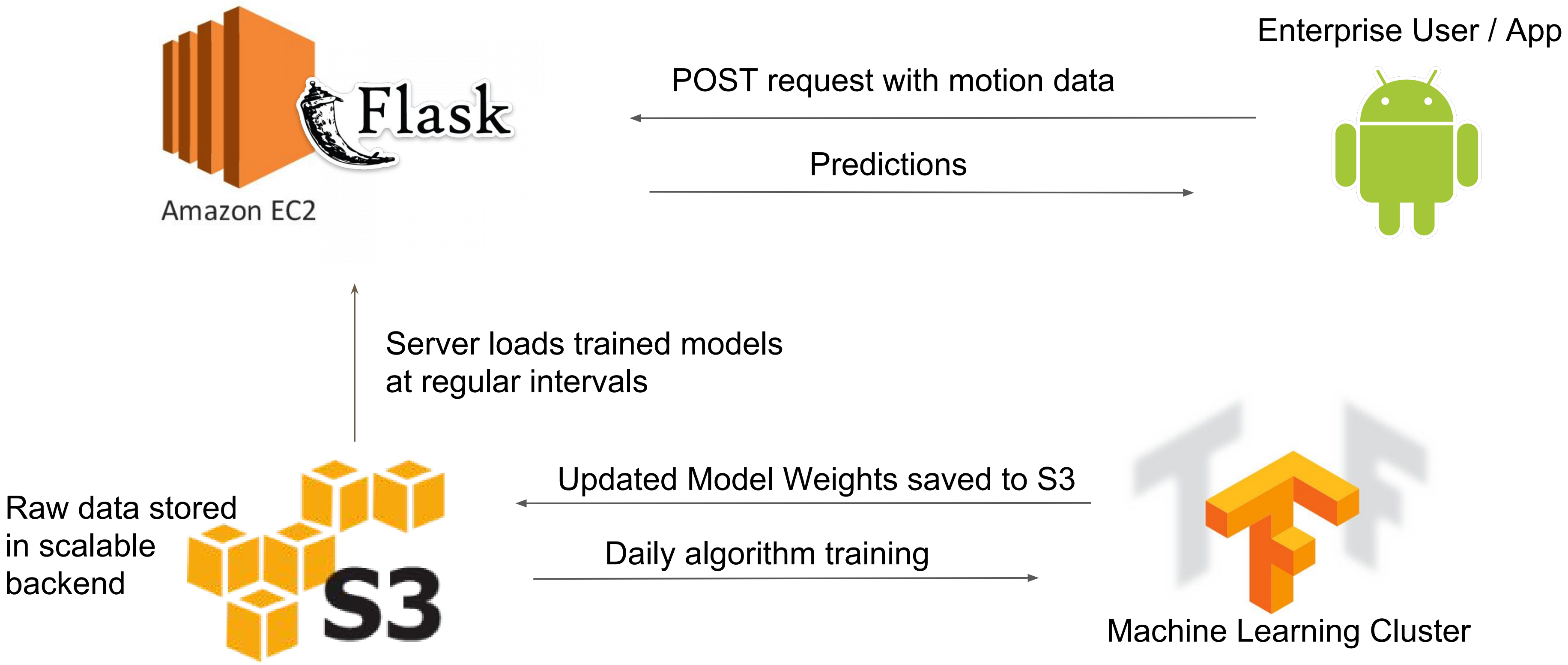
REST API

For enterprise consumers

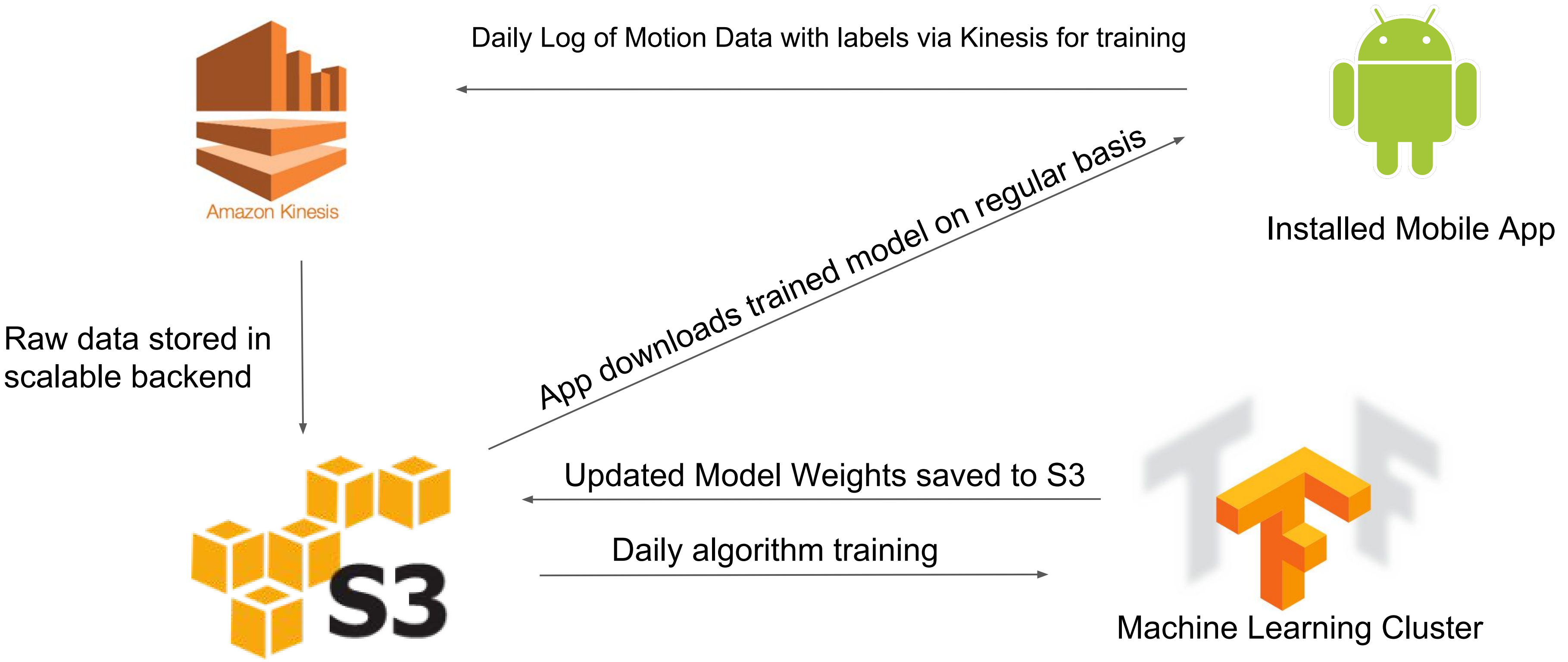
On-Device

For consumer app with high frequency prediction

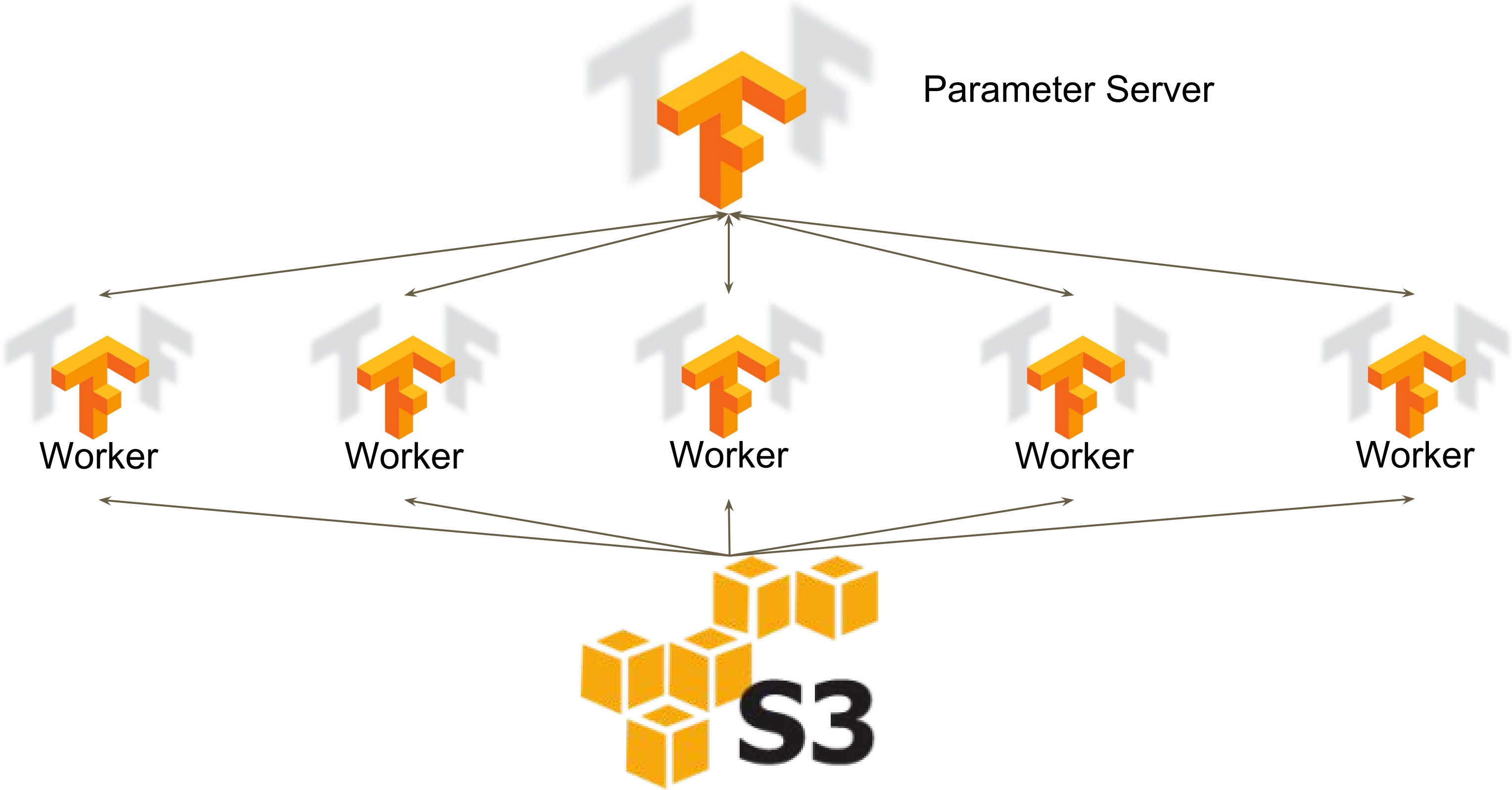
Deployment Architecture: REST API



Deployment Architecture: Model On-Device



Deployment Architecture: Distributed TF Training



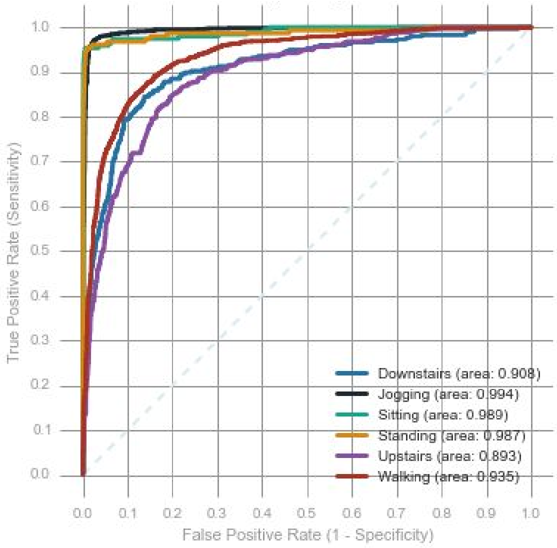
Future Works

- More contextual features such location, time
- More sensors such as rotational vectors, environmental sensors, GPS
- Working with unlabelled dataset and unsupervised learning

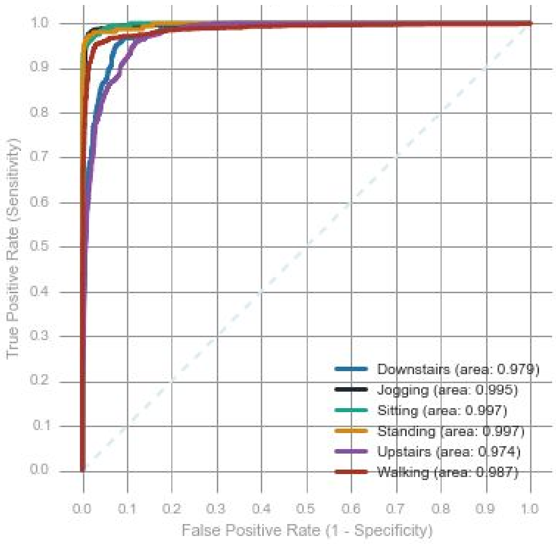
Thank You

Appendix: Model Comparison (ROC Curves)

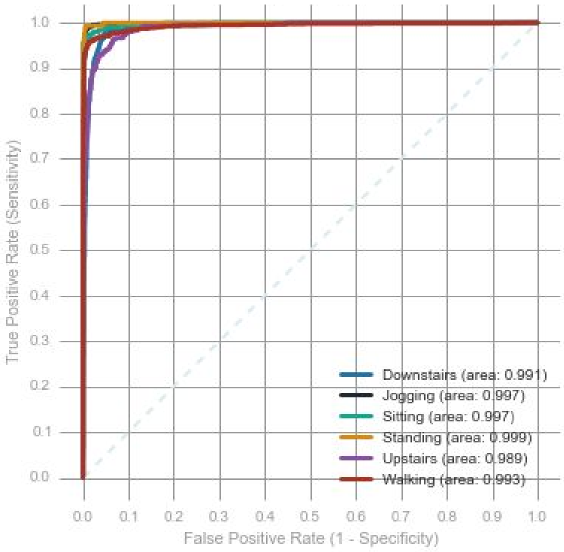
Logistic Regression



Multilayer Perception

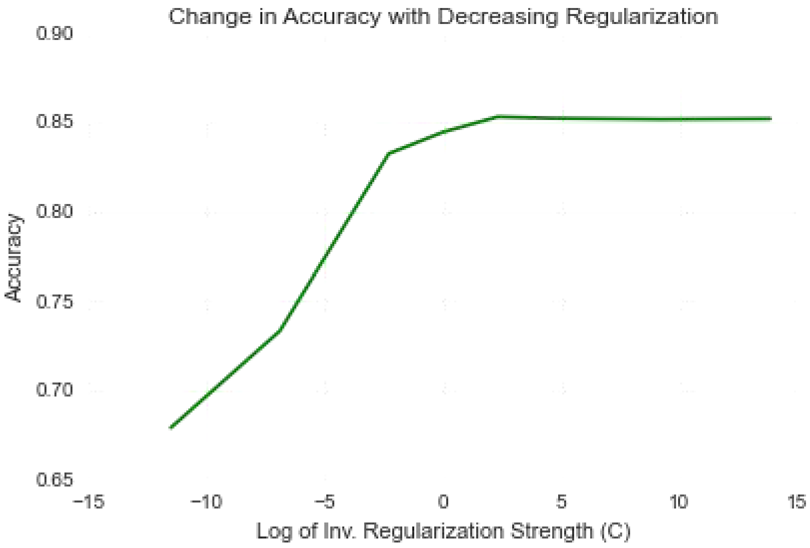


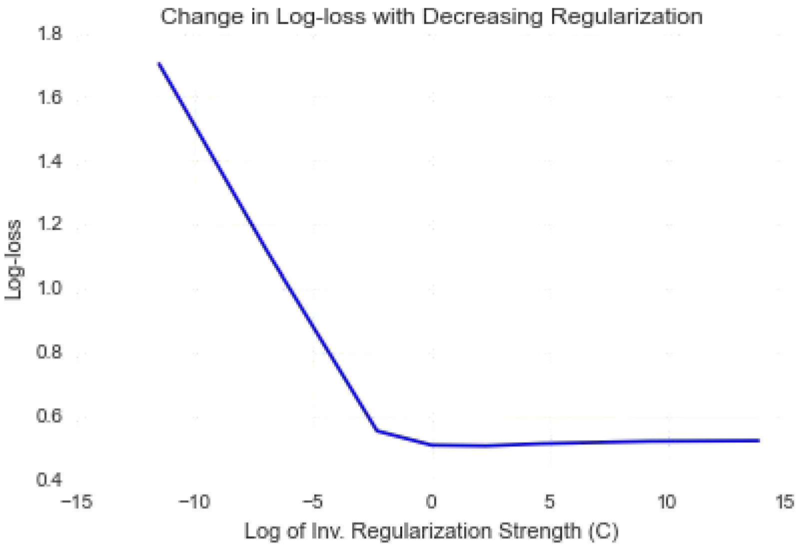
Boosted Trees



Appendix: Parameter Tuning

Logistic Regression





Appendix: Parameter Tuning

XGBoost