# Tejas Lodaya

https://tejaslodaya.com
Email : tejas.lodaya1@gmail.com
Mobile : +91-8892-246-348

ML Engineer with 3+ years of hands-on experience in Python, Scala, R and Go. My work lies at the intersection of machine learning and computer science to build end-to-end scalable ML solutions.

## Experience

- **Swiggy** — Bangalore, India
  *Senior Machine Learning Engineer* — *May 2019 - Present*
  - Built end-to-end platform to deploy deep learning models at scale using tensorflow serving. Achieved latencies of 15ms at peak 10k RPS using sidecar containers and gRPC network calls
  - Maintaining and operating ML platform with over 5k feature jobs and ~150 scala models in production
  - Streamlined visibility and monitoring of ML models by feature pipeline quality checks, alerts on data drifts, erroneous models and infrastructure failures. Reduced TAT for model failures from 1 day to 2 hours
  - Developed real-time and end-of-ride map matching algorithm to snap driver GPS pings to the underlying road network. Powering accurate driver payouts, missing road detection, ETA predictions and order assignments
  - Architected map-reduce style spatial querying and data manipulation engine for massive storage of point, line and polygon data. Provides real-time, cost-effective and performant solution for efficient OLAP queries
  - Built a low-throughput, high-latency prototyping platform to serve python-based models at scale. Enables data scientists to conduct experiments and validate hypothesis without rewrite in high-performant scala/tensorflow
  - Reduced AWS cost by 25% as part of operational excellence charter by optimizing spark jobs (vectorizing udfs, broadcast joins, partitioning strategies) and clusters (rightsizing executors, spot nodes, heterogeneous machines)
  - Deployed smart payments model into checkout springboot service with peak throughput of 2mn requests per day. Generated real-time features from the order flow and wrote JUnit cases for compile-time testing
  - Built performance testing framework to measure how fast models can fetch features and produce results when deployed in critical software systems. Helps gauge whether model is production-ready and within latency budgets
  - Worked closely with data scientists to onboard multi-objective models optimizing for competing metrics – UE,CX
  - Involved in design and solutioning of platform's automated retraining capability, model orchestration framework, centralized feature store and monitoring and alerting framework
- **Freshworks** — Chennai, India
  *Machine Learning Engineer* — *Oct 2018 - May 2019*
  - Migrated legacy codebase from open-source redis cluster to enterprise redis labs, reducing burn-rate by $250k per year. Implemented memcached to further increase key-fetch rate and reduce latency
  - Architected database model for storing normalized term frequency and document frequency across articles, achieving O(1) read and write speeds
  - Built APIs for exposing tf-idf ranking model to end customers through chatbot. Also integrated diverse use-cases like smalltalk, open-domain question answering, gibberish detector and custom intent detection engine
  - Worked on language agnostic spell-correct microservice achieving average search complexity of O(1), at the cost of pre-calculation time and storage space of $n$ deletions
- **Noodle.ai** — Bangalore, India
  *Associate AI Engineer* — *Jan 2017 - Sep 2018*
  - Built and orchestrated demand forecasting ecosystem for real-time consumption (using R). Wrote DAGs using Airflow as the workflow schedule system to run batch jobs
  - Worked on scaling compute by employing SIMD on N cores using parallel backends like doSnow, doParallel in R
  - Developed a proprietary ensemble modelling technique consisting of various models such as arima, xgboost, croston, prophet to capture heterogeneity of various timeseries

## Education

- **PES Institute of Technology** — Bangalore, India
  *Bachelor of Engineering in Computer Science; GPA: 9.00/10.0 (Top 5%)* — *2013 – 2017*

## Open source projects

- **Variational Recurrent Autoencoder**: Unsupervised, feature-based timeseries clustering algorithm in pytorch
- **Troop**: Simple library to perform chunkwise processing on data.frame across multiple cores of a single machine using SNOW clusters with a low memory footprint