Sungjin Lee (Jin)

T: 646-288-4859 | Email: sl4609@columbia.edu
Linkedin: linkedin.com/in/lsjhome/ | Github: github.com/lsjhome

TECHNICAL SKILLS

- Data Science: Python, R, numpy, pandas, scikit-learn, tensorflow, keras, pytorch, matplotlib, plotly
- Data Engineering: ETL, Spark, Mysql, Redshift, BigQuery, AWS EMR, EC2
- General Engineering: Docker, Git, HTML/CSS, Vim, Linux

EDUCATION

Columbia University — **New York, NY**
Master of Statistics — *Sep 2019 – Current*

- Probability, Inference, Linear Regression, Statistical Computing, Statistical Machine Learning, Intro to Databases, Computer Systems for Data Science, Linux Project
- Assistant Director, Academic department of Columbia Statistics Club, SQL seminar instructor

Pusan National University — **Busan, South Korea**
Bachelor of Economics, minor in Mathematics — *Mar 2006– Feb 2015*

- Linear Algebra, Calculus, Inferential Statistics, Combinatorics, Econometrics
- Merit Scholarship, Full Scholarship

WORK EXPERIENCE

LeaseLock — **Los Angeles, CA**
Data Science Intern — *Mar 2020 – Current*

Predicting the key performance indicators for risk platform

- Built hundreds of time series models for sales forecasting using *prophet, pandas, Python, airflow, s3, redshift, Docker*
- Implemented data enrichment module and added it to ETL jobs using *Python, census, airflow, s3, redshift*
- Performed Exploratory Data Analysis on new tables using *plotly, pandas, Python*

Smartforecast — **Seoul, South Korea**
Data Scientist — *Mar 2018 – Mar 2019*

Customer Scoring for insurance company

- Built classification machine learning models that score customers based on the probability of buying company's insurance for better and more optimized marketing, using *scikit-learn, xgboost, catboost, pandas*
- Evaluated machine learning models extensively using stratified cross validation, precision, rank metrics and observed 3.2x lift
- Developed a RESTFUL microservice and dockerized for the production environment using *Docker, Python, and Flask*

Youtube Data Crawling

- Developed a scalable crawler that crawls Youtube video metadata, comments and stored data into database, using youtube api, *Python, request, loggings, pymysql, multiprocessing, deque*
- Improved the crawler's performance by 20x by utilizing multithreading

Robot Journalism

- Managed system that generates stock market article on weekly basis for economy news, using *pymysql, aws, matplotlib, nltk*
- Recommended to buy or sell stocks for swing trading based on news article, using NLP, such as keyword extraction and sentimental analysis

Entropylab — **Seoul, South Korea**
Data Scientist — *Apr 2017 – Nov 2017*

Horse racing prediction

- Hired as the first employee and played a critical role in establishing company name, product name, recruiting, workflow
- Automated data crawling from public API using Python and shell scripts, using *crontab, CentOS*
- Applied various machine learning algorithms such as Rainforest Tree, XG Boost, Logistic Regression, Catboost to forecast horse racing results, using *scikit-learn, pandas, numpy, catboost, xgboost*
- Published horse racing expert "Hangura" as an app on Google Play, which outperformed all professional Korean horse racing experts in terms of comprehensive accuracy on the horse racing web community.
- Google play link: [https://bit.ly/2VvegW7] (service stop)

Certiware — **Seoul, South Korea**
Data Scientist — *Dec 2016 – Mar 2017*

Recommendation system for online shops

- Designed recommendation system ERD for online shops
- Developed test data generator for quality assurance, using *Python*

PROJECTS

A dog breed Image classification model [https://bit.ly/2NdDeU1]

- Built human and dog detector that returns resembling dog breed or dog breed with data augmentation, transfer learning, using *PyTorch, resnet*
- Used 13,233 human images for human detector and 8,351 dog images for dog breed classifier (133 breeds, 86% accuracy for breed classification)

TV script Generator [https://bit.ly/2NgNWsJ]

- Built Seinfield TV script generator
- Built word embedding layer for creating word vectors before LSTM layer, using *PyTorch*