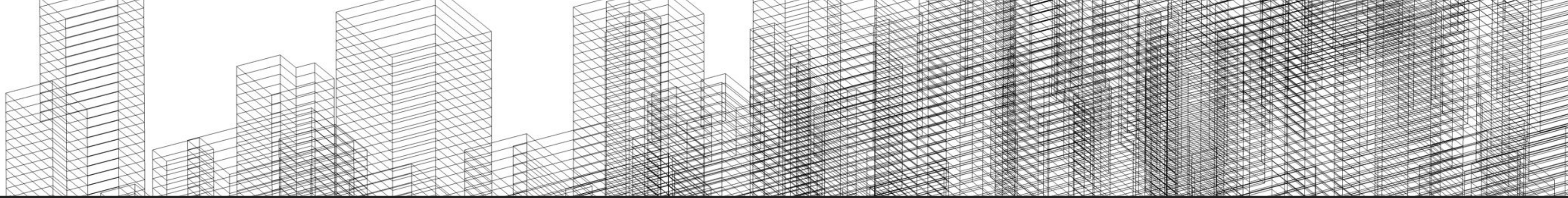
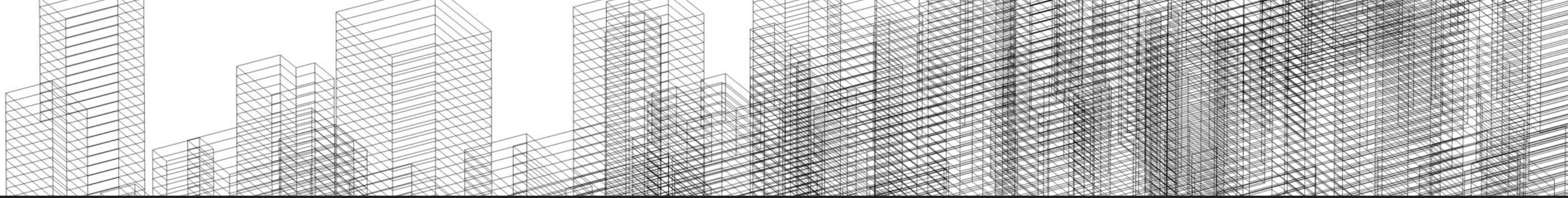


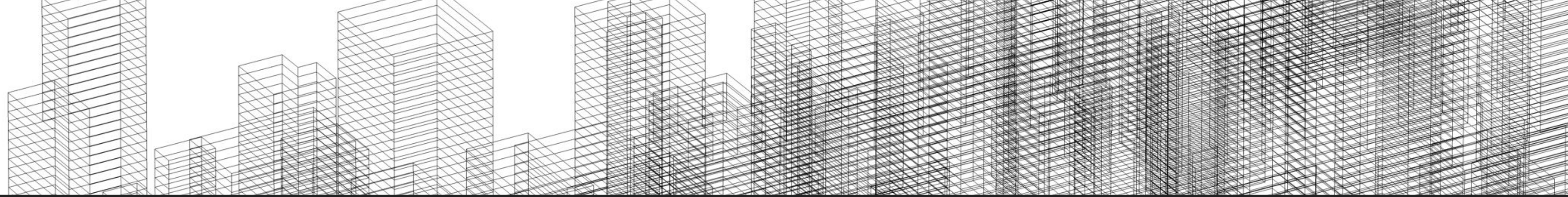
App-store Project

Mid-term Checkpoint
Eileen Zhang



Highlights

- Exploratory Data Analysis to explore the relationship between variables and determine which subset of variable to use.
- Create several new predictors based on initial EDA
- Built linear regression, random forest and gradient boosting models to predict the user ratings using a set of attributes. (performed 5 fold cross validation on all of the models to prevent overfitting)
- The gradient boosting model has R squared of 58% which improves 50 % compared to the initial simple linear regression model.
- The model could predict new rating based on user input.



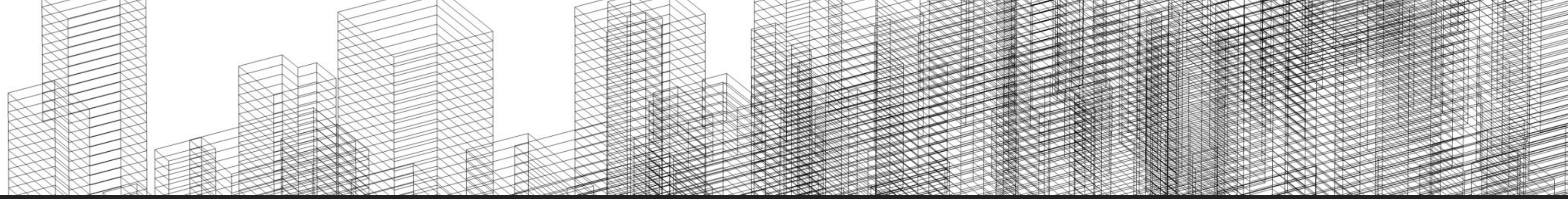
Review Progress

Epic 1: Exploratory Data Analysis to explore the relationship between variables and determine which subset of variable to use. (user input related)

- Story 1: Merge two data sets
- Story 2: Data cleaning, check NA values and duplicates.
- Story 3: Distribution of each variables, check multicollinearity and outliners

Epic 2: Feature Engineering

- Story 1: Variable transformation



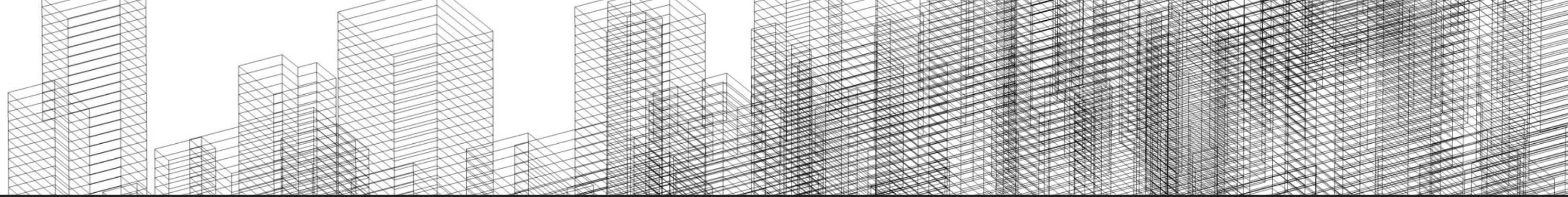
Review Progress

Epic 3: Model Building to predict the number of reviews and ratings using a set of attributes.

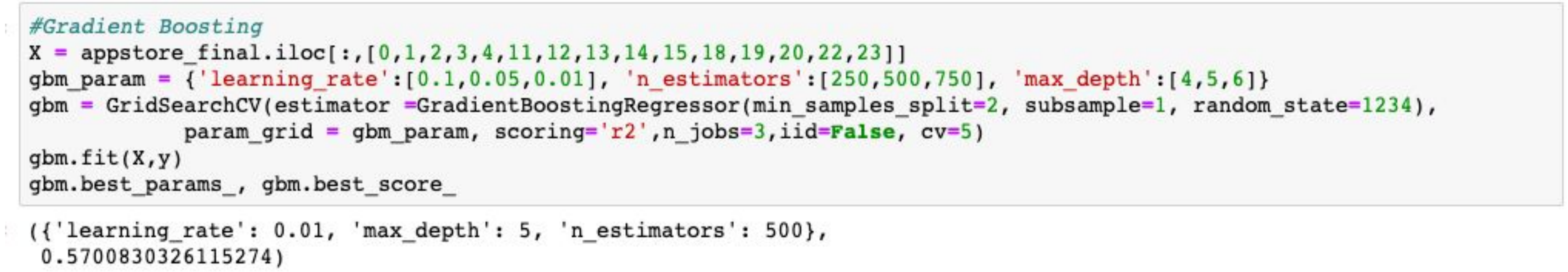
- Story 1: Split the data set to training and test set
- Story 2: Run a simple linear regression model to use its R square as the naive baseline. Check variable importance and residual plot to determine what advanced model to use.
- Story 3: Run Neural Network, Random Forest Model to predict the number of reviews and ratings. Making sure the model hit the performance metrics.
- Story 4: If models above don’t perform well as expected, try more such as gradient boosting, GAM, etc.

Epic 4: Build Pipeline

- Story 1: Set up the environment

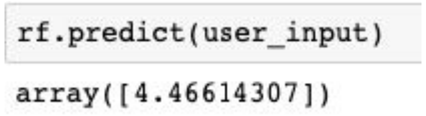


Demo



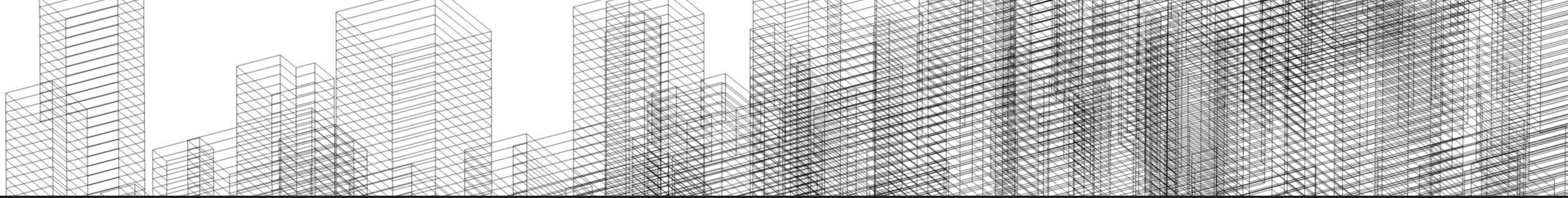
```
#Gradient Boosting
X = appstore_final.iloc[:,[0,1,2,3,4,11,12,13,14,15,18,19,20,22,23]]
gbm_param = {'learning_rate':[0.1,0.05,0.01], 'n_estimators':[250,500,750], 'max_depth':[4,5,6]}
gbm = GridSearchCV(estimator =GradientBoostingRegressor(min_samples_split=2, subsample=1, random_state=1234),
            param_grid = gbm_param, scoring='r2',n_jobs=3,iid=False, cv=5)
gbm.fit(X,y)
gbm.best_params_, gbm.best_score_
```

```
({'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 500},
 0.5700830326115274)
```



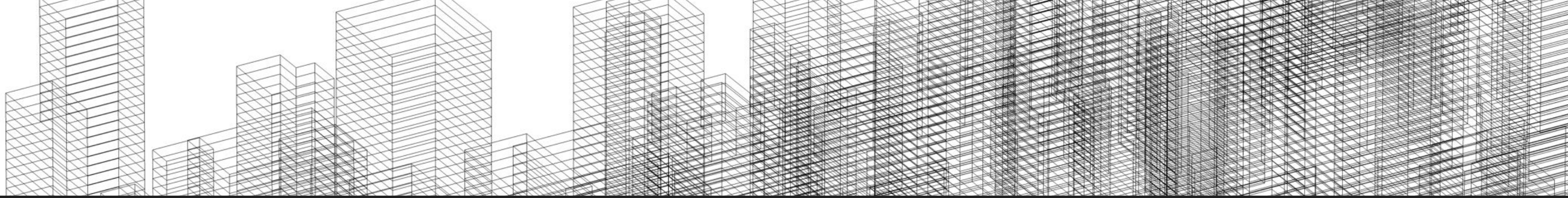
```
rf.predict(user_input)
```

```
array([4.46614307])
```



Lesson Learned

- When doing prediction models, if the model performance is always below expectation, it is worth to reconsider the scope or the modeling type of the project.
- Setting up the configuration and environment at the very beginning is much more convenient compared to setting them up halfway. Although, these basic stuff may take some time at first, once we have a solid foundation, everything built on top of it will be very smooth and easy.



Recommendations

- Epic 4: Build Pipeline
  - Story 2: Move algorithm from local to AWS
  - Story 3: Run and save the trained model
  - Story 4: Set up the data pipeline (e.g. RDS)
- Epic 5: Web App Development to build interactive features that allow users to enter their data to get predictions.
  - Story 1: Writing the backend using Flask app.
  - Story 2: Web page design, basic layout
  - Story 3: Create interactive page allow the user to enter the attributes of their apps, attributes should be corresponds to the parameters used in the models. (8 points)
  - Story 4: For certain attributes, enable users to click on instead of simple text entry for diverse user experience.
  - Story 5: Predict and show the number of reviews and ratings for the specific file on the same page in 20 seconds.
  - Story 6: Aesthetic design, color palette, font, format details.
- Epic 6: Running the application
  - Story 1: Test run the app to evaluate the performance.