Kashif Khan: 120196096 | Eric Juta: 120155509 | Yonathan Maalo: 120591833

Titanic Competition

Overview

**Kaggle Leaderboard Name:** kashif

**Score:** 0.818

**Time Taken:** 3 Weeks

**Method:** SimpleCart classifier available from weka api, implemented in java, with pre-processing and assumptions.

**Time to train model & test:** Very fast.

**Tasks:** Eric: Scripting & Research | Kashif: Scripting, Research and Processing Data | Yonathan: Research

Approach

* **Java Application:** The java application utilised all the weka tree classifiers available, and from this we kept looking at the number of correctly classified instances. We would train the classifier from the csv file and then test its accuracy using cross validation and percentage splits. The application then would load and print out the result of classifying a test instance on the survived attribute. We would then upload the outputted csv file and upload it into Kaggle to validate.
* **Pre-processing:** We removed the cabin, ticket, embarked and name fields in the initial tests. We then converted the Pclass and survived fields so that the weka classifiers would immediately recognise as nominal attributes rather than numeric; yes and no for survived, and a b c for passenger classes.
* **Issues:**
  + **Missing Data:** Weka showed that by using a “replace missing value” filter on the age attribute, that there was a decrease in accuracy for all classifiers, therefore it was not used.
  + **Title Field:** After consistently getting average scores of, we realised that we could extract more data from the name field, referring to the titles of passengers. We created a new attribute – title – and then reran classifiers, where the test accuracy improved. Because there were so many different types of titles, we concluded that this was overfitting to the train data, and created a new title named “noble” – dr., don., sir., capt. Etc. This drastically improved our best score to .809
  + **Families:** Looking at the train data in excel, we an assumption that boosted our score: Searching for cases where very young children who died, both parents would have died and vice versa. This pushed our score to 0.818!

Bike Sharing Competition

Overview

**Kaggle Leaderboard Name:** rej156

**Score:** 0.40827

**Time Taken:** 3 Weeks

**Method:** Gradient Boosted Trees model. Trained separate models to predict the number of registered users and casual users. Combined the predictions of both models. Feature Engineering also done.

**Time to train model & test:** Train time – ~2 minutes| Test time - ~2 minutes

**Tasks** Eric: Scripting & Research | Kashif: Research | Yonathan: Scripting & Research

Approach

**Python Application:** The application utilised a library called Graph Lab Create, which includes a fork of XGBoost to produce a fast Graph Boosted Model. We felt the main point of this task was the feature engineering. We tried splitting the data in a number of ways and visualised the result to see if this feature revealed any differences in the count. If there were differences, we kept this feature. We trained two separate models, to predict the registered count and the casual count and combined these models to give a more accurate overall count. Some tuning of the hyperparameters of the Gradient Boosted Trees model was also needed.

**Dealing with:** Feature Engineering – Split date-time into year, month, day, hour and weekday. We also split the date into a Sunday column. We also included a day-part feature, grouping the hours into six hour blocks.

**Hard parts & solutions:** Feature Engineering/Pre-processing. Split the data and see if there are variances in the number of bikes being rented according to any split. Make the relevant splits of data a feature.