Topics in Computer and VLSI Project

(Group 9)

Student:
LE VAN DUC
2016-27885

Student:
Ali Almokhtar
2015-30846

[Libraries using and instructions to run]

- **Libraries**:
    - Pandas v0.20.1
    - Scikit-learn v0.18.1
    - Scipy v0.12.1
    - Numpy v1.12.1
    - CSV, Math, Random
- **Instructions to run:**
    - Regression:
        - Basic model:
            - Python regression.py : show the validation result.
            - Python regression.py test path_to_test_file: for testing.
        - Optimization model:
            - Python regression_opt.py : show the validation result.
            - Python regression_opt.py test path_to_test_file: for testing.
    - Classification:
        - Model 1:
            - Python classifier1.py : show the validation result.
            - Python classifier1.py train : to train from the beginning.
            - Python classifier1.py test path_to_test_file : for testing.
        - Model 2:
            - Python classifier2.py : show the validation result.
            - Python classifier2.py train : to train from the beginning.
            - Python classifier2.py test path_to_test_file : for testing.

A. [Price prediction]

Preprocessing data:

- We use pandas to read csv training data file.
- To make the model more accuracy, we dropped many columns which has more than **20%** of missing values, like “Alley” column has 1186/1260 missing value, “Fireplace Qu” has 599/1260 missing values, “Pool QC” has 1253/1260, “Fence” has 1013/1260 and “Misc Features” has 1217/1260.
- After drop too many missing values columns, we filled remaining missing values columns with the mean value of each column => this is more reasonable than fill with 0 or drop.
- We also used hot encoding library from SKlearn to encode categorical features using a one-hot aka one-of-K scheme. After using “from sklearn.preprocessing import OneHotEncoder”. LabelEncoder, StandardScaler . We got a new shape **(1260, 269)** from (1260, 79) original shape.

1) File regression.py

- We used Linear Neural Network to predict the house price. The architecture of the model consists of: 1 Input Layer with number of nodes = the features of input data (269 after One Hot Encoder), 1 Output Layer with 1 nodes = the predicted sale price.
- Loss function with 1 training example: $L(w) = ½ * (w.T * x - y)^2$ with w = weights, x = training example, y = real house price.
- Gradient: $\frac{\delta L(w)}{\delta w}$ = (w.T * x - y) * x.
- Training with gradient descent: w = w - learning_rate * gradient
  = w - learning_rate * (w.T * x - y) * x
- We used 3000 iterations with 200 epochs. We get around 60,000 iterations in total. The learning rate chosen = 0.0002 with batch size 100. The predicted output compared to the real price is shown below (with 10 top values)

Predicted	Real value.
164477	134000
157664	110000
185081	284000
132671	97000
146387	119000
185310	175500
144647	78000
201958	200000
154970	125500
203934	320000

Also the error matrices are shown below:

Bias error	-6361.58
Maximum deviation error	852964.54
Mean absolute deviation error	50814.73
Mean squared error	10951732664.79

2) file regression_opt.py

We used dimensionality reduction algorithms (PCA) from sklearn dimensionality and sklearn.decomposition . PCA reduce the data. After reduction the new shape: (1260,40).

We run also the same model using the data set after reduction. Our model had performed better because it had only related data which was used to feed our network with. As you can see below,

Predicted	Real value
152235	134000
147842	110000
171194	284000
108149	97000
133409	119000
175413	175500
129982	78000
186416	200000
141208	125500
188205	320000

and the error matrixes as shown below. We got slightly better performance.

Bias error: 2806.27

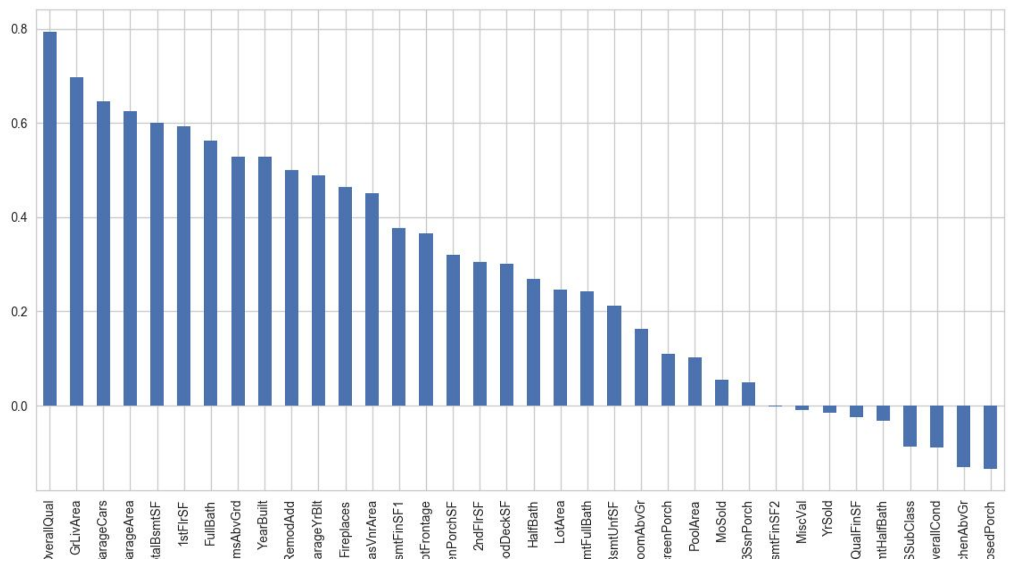
Maximum deviation error: 492729.34

Mean absolute deviation error: 47390.00

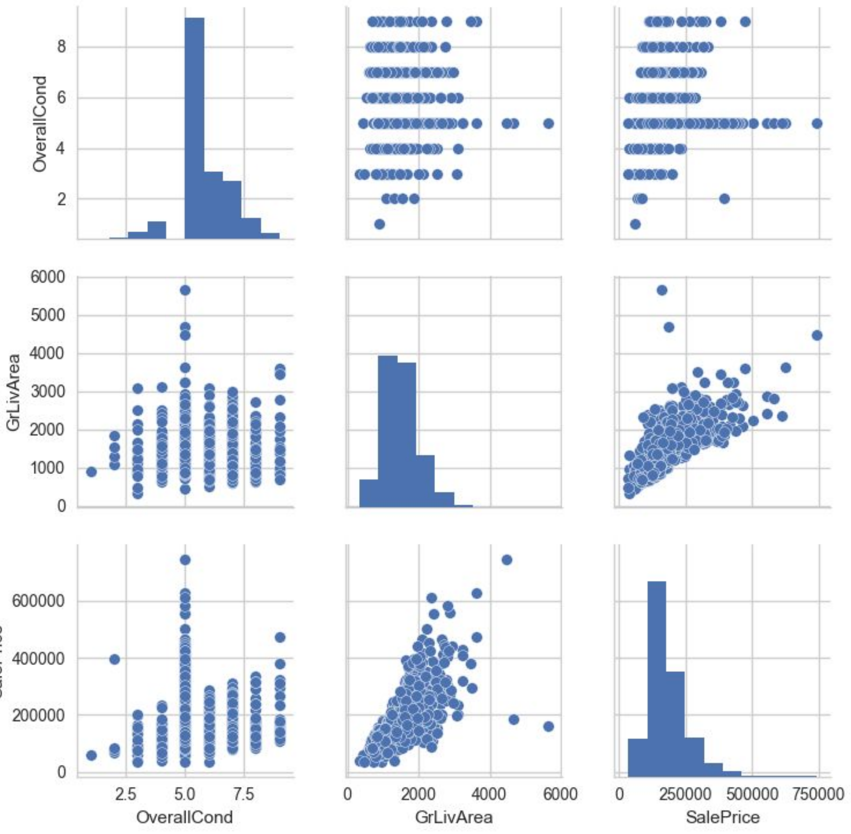
Mean absolute deviation error: **6102452566.22**

3. Show which variables are comparatively more related to the house price:

The graph below shows correlation between SalePrice and the different features. As we can see , the two top (OveralQual , GrLivAreaa) has the highest correlation with sales price.



Plot Saleprice as a function of the two top correlated features.



B. [Price range classification]

1. **Program 2 classification models that can classify the house price into 2 classes**
   - Pre-processing training data:
     - Similar to the pre-processing data in Regression part.
     - We also do the followings:
       - Read the csv file by pandas library.
       - Remove columns (features) which have the number of missing values more than 20%.
       - With remaining columns which still have missing values but less than 20%, we fill the missing data by mean value of each column.
       - Then, the data is scaled to [0-1] range to make the training faster and more accurate.

        - The output classes target: class 0 if price < 160,000 and class 1 if price >= 160,000.

A) File classifier1.py : Neural Networks model

- Architecture:
    - 1 Input layer: nodes = number of data features. In this case we will drop all categorical features => number of data features = 36.
    - 1 Hidden layer: we choose number of nodes of hidden layer = 10.
    - 1 Output layer: 2 nodes corresponding to 2 output classes.
- Explain the algorithm:
    - We choose the Neural networks model because NN model can be used efficiently in the classification problem. Otherwise, we can easily update the model by changing hidden layer to improve the model.
    - Loss function: $\mathbf{L(w)} = \sum_{i=1}^{N} \frac{1}{2} * (\mathbf{output}^{(i)} - \mathbf{target}^{(i)})^2$. With output$^{(i)}$ is the output after running 1 example (i) forward to the model. Target$^{(i)}$ is the real class (0 or 1) of the training example.
    - Use sigmoid function for activation function. The number of training data is not many and the data is scaled to [0-1] range so we can use sigmoid function and not worry about the saturation.
    - Use back propagation algorithm to train this model with **learning rate = 0.001** and **1000 iterations** with each epoch. For each iteration, we train with stochastic mini-batch training data with **batch size = 100** to reduce training time.
    - Feedforward algorithm to be used to compute the output of the model for back propagation algorithm and to produce prediction training and testing.
    - To validate the accuracy of the model, with each epoch, we randomly split data set to training data (**80%**) and validating data (**20%**) and evaluate the model accuracy on the validate data.
    - To rerun the model with additional test set, we store the training weights in file and reload if run in testing data.
- The in-sample error of the model:
    - In-sample error rate ~ 48.33%.

B) File classifier2.py: Logistic Regression model

- Architecture:

    - Model formulation: $\frac{exp(b0 + \sum_{i=1}^{L} biXi)}{1.0 + exp(b0 + \sum_{i=1} biXi)}$ with L = number of features
    - b0 =bias, bi = learning coefficients.
- Explain the algorithm:
    - We choose Logistic Regression algorithm because this algorithm can be used to classify a binary output. And this algorithm is quite simple to train and implementation.
    - We use Stochastic Gradient Descent to minimize (train) the model. At each iteration, we loop through all training data and compute the gradient descent:
    - Gradient descent: **delta = error * yhat * (1.0 - yhat)** with **error = (y - yhat)**, yhat = prediction made by the model, y = real target result.
    - Then, update coefficients by: **b = b + learning_rate * delta**. Learning rate chose = 0.001. The number of iterations = 1000.
    - To validate the accuracy of the model, we randomly split data set to training data (80%) and validating data (20%) and evaluate the model accuracy on the validate data.
    - To rerun the model with additional test set, we store the training coefficients in file and reload if run in testing data.
- The in-sample error of the model:
    - In-sample error rate ~ 36.41%.

2. Which variables are comparatively more related to the house price class:

- The features related to house price class are similar to the features related to the house price: OveralQual, GrLivArea, ...

3. Improve the performance of one model

- I will improve the performance of Neural Networks model by 2 methods:
    - Dimensional reduction to make the model not overfitting with training data.
    - Change the number of nodes of the hidden layer and use Cross-validation selection to choose which hidden layer number has the best accuracy on validation set. We applied this method because the hidden layer number effects to the number of learning weights of the model. Changing this will change the learning capacity of the model.

- Dimensional reduction:
    - Library: sklearn.decomposition.TruncatedSVD is used.
    - To see which dimensional reduction will have the best performance, we test by Cross-validation selection with n_components of dimensional reduction from 30 to 70 (step = 10).
    - After validating, the best n_components is 40.
- Change hidden layer number:
    - After getting the n_components of dimensional reduction.
    - We change the hidden layer number from 10 to 50 (with step 10).
    - Using Cross-validation selection, we got the best hidden layer number is 40.
    - The in-sample error rate ~ 23%.