Machine Learning

*Analyzing problems, designing a machine learning solution, implementing ML algorithms, and evaluating data sets

1st Saksham Saxena
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

2nd Rushabh Doshi
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

3rd Luke Beaulieu
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

4th Utkarsh Patel
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

***Abstract*—This paper presents the analysis and evaluation on data sets using machine learning techniques. Our method in this assignment consist of working with three data sets and implement ML algorithms on them to understand more about the field of machine learning and its applications in the real world. The three three data sets worked on in this project was Boston housing, breast cancer, and Forest Covertype.**

***Index Terms*—introduction, methodology, result, discussion, conclusion**

I. Introduction

The goal of this final project is to show that we have learned something in the class. It is an opportunity for us to explore ideas that we have see in the lectures and assignments and extend them. This project is a first steps towards research in machine learning or its applications. We will analyze problems, design a machine learning solution, implement ML algorithms, and evaluate them on three data sets (one for classification and one for regression from Small Data Sets and one data set either classification or regression from Large Data Sets).

II. Problem and Data sets

A. Breast Cancer, logistic regression

Breast cancer data set is a classification data set. The output variable of the cancer data was either malignant or benign .

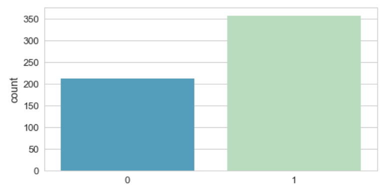


Fig. 1. 0: Malignant 1: Benign

Since this is a classification data set, it was important to go back and look at which methods I could and could not use to work with this data set. I chose logistic regression model in this case. The idea of logistic regression is that there is an optimal decision boundary that separates the two classes of cancer. From class, we learned methods to obtain that optimal decision boundary using cost function.

For the given training data, we want to find parameters Θ that are most likely by maximizing L(Θ). This is the maximum likelihood estimator.

Cost function

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Cost}(h(\mathbf{x}^{(i)}), y^{(i)})$$

$$\begin{aligned} L(\boldsymbol{\theta}) &= p(y^{(1)}, \ldots, y^{(n)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}; \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})^{y^{(i)}} (1 - \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}))^{1-y^{(i)}} \end{aligned}$$

Fig. 2. maximum likelihood estimator function

Knowing the likelihood function (above) for this given problem, we look for such Θ that maximizes the probability of obtaining the data we have. To find the Θ, we have an optimization algorithm to find that Θ.

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Fig. 3. gradient descend where n is the learning rate

Going through different learning rates and iterations, this leaves us with an optimal line that separates the two classification of breast cancer. *n is the learning rate and it determines how big of a step we get to the minimum point in the gradient descend. -Too big of a learning rate causes drastic updates which leads to divergent behavior. -Too small of a learning rate and it may take a lot of iterations to get to the minimum point.

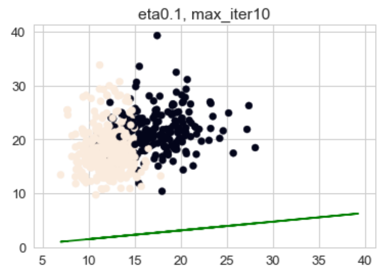


Fig. 4. Small learning rate and iteration

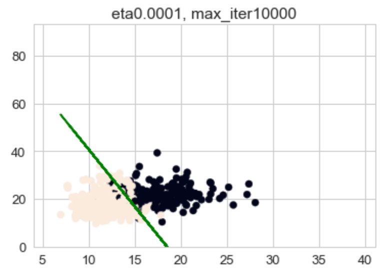


Fig. 5. Good learning rate and iteration

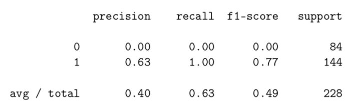
Although I did not have enough time to find the numerical of the errors and accuracy, that is what I would have liked to do if i had more time.(doable using train and test using a 40:60 split model)

The logistic model optimally draws the line between two classes of data, so these results (graphs) do make sense.

B. Breast Cancer, Support Vector Machine

Support Vector Machine or SVM is another method for classification problems such at breast cancer. This works efficiently on datasets that are linearly separable, and so what SVM attempts to do is separate the data by a 'separating hyperplane'. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. [wiki].

The methodology for SVM was to setup the data frame and use a train test 40:60 split. Then using the support vector classifier, or SVC model, I went to predict the accuracy of the model, how well it maps new data or examples for this data.

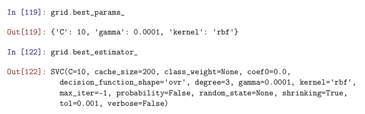


```
             precision    recall  f1-score   support

          0       0.00      0.00      0.00        84
          1       0.63      1.00      0.77       144

avg / total       0.40      0.63      0.49       228
```

Fig. 6. Prediction with no adjustment

Obviously this was not a good prediction without training, adjustment, and normalizing the data. The model is fairly inaccurate.

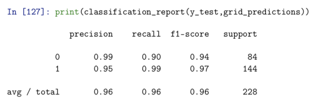
For the parameters of the SVC model, we want to use **grid search** to find the best parameter, specifically C and Γ. Grid search runs the same loop with cross-validations to find the best parameters. Once it has found the best combinations, it runs fit on all the data passed to build a single model using the best parameter setting.



```
In [119]: grid.best_params_

Out[119]: {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}

In [122]: grid.best_estimator_

Out[122]: SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf',
            max_iter=-1, probability=False, random_state=None, shrinking=True,
            tol=0.001, verbose=False)
```

Fig. 7. best parameters & estimator

You can pull out the best parameters and estimators from grid search. With the adjusted parameters and normalization of the train test split data, we get a better accuracy for the model.



```
In [127]: print(classification_report(y_test,grid_predictions))

             precision    recall  f1-score   support

          0       0.99      0.90      0.94        84
          1       0.95      0.99      0.97       144

avg / total       0.96      0.96      0.96       228
```

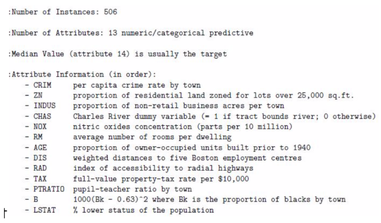
Fig. 8. Prediction with adjustment + normalization

The SVC model does make sense in terms of mapping an example to a class with an accuracy of 96%. With more time, I would extend this project to set up more classification models, including Neural Networks and K-Mean Clustering. With these different models, an obvious thing to do is compare and contrast the different models. An entire different report on the comparison would include errors margins, efficient margins and more data to describe the different techniques used.

On the side, this project could extend to doing research and problem solving in other areas aside from the data sets we worked on. I would have loved to do a research project on how students past grades affect future classes, but that is for the future and a different time.

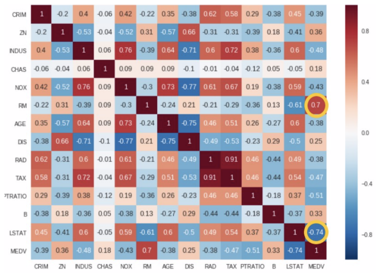
C. Boston Housing , linear regression

The Housing dataset contains information about houses in Boston but the information is from early 1990s. The dataset is available in the scikit learn library. It contains 506 samples and 13 feature sets including features like number of rooms, age of the house and per capita crime rate of the town. The objective is to predict the price based on the price data available for the given samples.



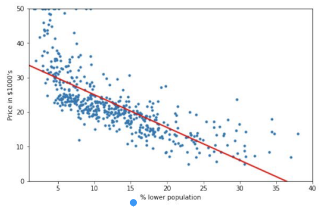
```
:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive

:Median Value (attribute 14) is usually the target

:Attribute Information (in order):
    - CRIM     per capita crime rate by town
    - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
    - INDUS    proportion of non-retail business acres per town
    - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
    - NOX      nitric oxides concentration (parts per 10 million)
    - RM       average number of rooms per dwelling
    - AGE      proportion of owner-occupied units built prior to 1940
    - DIS      weighted distances to five Boston employment centres
    - RAD      index of accessibility to radial highways
    - TAX      full-value property-tax rate per $10,000
    - PTRATIO  pupil-teacher ratio by town
    - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
    - LSTAT    % lower status of the population
```

We started with analyzing the dataset to get a better understanding of the data before us. To get more information about the features, we used boston.DESCR function to get a detailed information about each of the feature. Before using the data, its always a good habit to preprocess the data and look for some missing values. Fortunately, there wasnt any missing value in our dataset. Next, we used the heatmap function of seaborn library to get the correlation matrix that measures the linear relationship of each feature with our target value as well as with all other features.

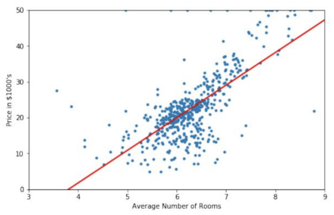


	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	-0.2	0.4	-0.06	0.42	-0.22	0.35	-0.38	0.62	0.58	0.29	-0.38	0.45	-0.39
ZN	-0.2	1	-0.53	-0.04	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
INDUS	0.4	-0.53	1	0.06	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
CHAS	-0.06	-0.04	0.06	1	0.09	0.09	0.09	-0.1	-0.01	-0.04	-0.12	0.05	-0.05	0.18
NOX	0.42	-0.52	0.76	0.09	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
RM	-0.22	0.31	-0.39	0.09	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
AGE	0.35	-0.57	0.64	0.09	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
DIS	-0.38	0.66	-0.71	-0.1	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
RAD	0.62	-0.31	0.6	-0.01	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
TAX	0.58	-0.31	0.72	-0.04	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
TRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
B	-0.38	0.18	-0.36	0.05	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
LSTAT	0.45	-0.41	0.6	-0.05	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
MEDV	-0.39	0.36	-0.48	0.18	-0.43	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

To fit a linear regression model, we selected those features which have a high correlation with our target variable MEDV(Price). By looking at the correlation matrix we can see that RM has a strong positive correlation with MEDV(0.7) where as LSTAT has a high negative correlation with MEDV(-0.74). The important thing to keep in mind here is to avoid those features with multi-co-linearity. As can be seen in the heatmap image, RAD and TAX have a correlation of 0.91. Lets consider an example to understand why we do so. Consider the simplest case where Y is regressed against X and Z and where X and Z are highly positively correlated. Then the effect of X on Y is hard to distinguish from the effect of Z on Y because any increase in X tends to be associated with an increase in Z. So based on these observations, we used RM and LSTAT as our features to build the Linear Regression model.



At this point, I ran an experiment to check whether the linear model improves based on my assumptions or not. I built the Linear Regression model using the scikit learn class. But



instead of one, I made two models and compared the results. In the first model, I just selected the two features which we got from analyzing the data before. In the second model, I selected all the 13 features to see whether it has any effect on the predictions. Even though I did get a better result based on RMSE for the first model, the difference in the RMSE values wasnt as big as I expected it to be. And this is something which I need to think about and try to understand the reasoning behind it.

D. *Forest Covertype*

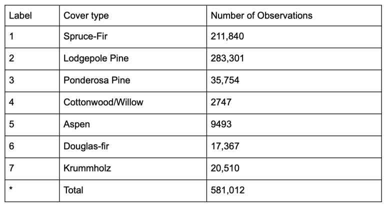
Dataset

The complete dataset is composed by 581,012 instances, where each observation (instance) corresponds to a 30 x 30 meters cell. For each observation, 12 measures (attributes) are given. However, two categorical properties (wilderness area and soil type) were binarized, in order to have a dataset composed only by numerical attributes. So, the final dataset is composed by 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables, given a total of 54 columns of data. Below, some description about these attributes is given:

- Elevation: Elevation in meters.
- Aspect: Aspect in degrees azimuth.
- Slope: Slope in degrees.
- Horizontal distance to hydrology: Horizontal distance to nearest surface water features.
- Vertical distance to hydrology: Vertical distance to nearest surface water features.
- Horizontal distance to roadways: Horizontal distance to nearest roadway.
- Hillshade 9am: Hillshade index at 9am, summer solstice (0 to 255).
- Hillshade Noon: Hillshade index at noon, summer soltice (0 to 255).
- Hillshade 3pm: Hillshade index at 3pm, summer solstice (0 to 255).
- Horizontal distance to fire points: Horizontal distance to nearest wildfire ignition points.
- Wilderness area: Wilderness area designation. 4 binary columns, 0 (absence) or 1 (presence).
- Soil type: Soil type designation. 40 binary columns, 0 (absence) or 1 (presence).
- Cover Type: Forest cover type designation. This is the attribute to be predicted (1 to 7).

Regarding the Cover Type attribute, the following distribution is observed in the whole dataset:

Random Forest

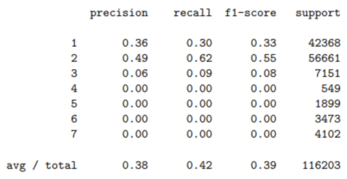


Label	Cover type	Number of Observations
1	Spruce-Fir	211,840
2	Lodgepole Pine	283,301
3	Ponderosa Pine	35,754
4	Cottonwood/Willow	2747
5	Aspen	9493
6	Douglas-fir	17,367
7	Krummholz	20,510
*	Total	581,012

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. We use sklearn.ensemble. RandomForestClassifier from scikit-learn package to do the classification.
ensemble.RandomForestClassifier(n_estimator = 100, max_depth = 2, random_state = 0) n_estimators stands for the number of trees in the forest. We use fit(X, y) to build a forest of trees from the training set (X, y) and predict(X) to predict class for X. Stratified Dataset Another method that was used was stratifying the dataset. When a dataset is very highly skewed with a particular label field. It can affect the performance of that model, because the dataset the model was trained on is not reflective of the dataset the model was tested on. When a dataset is stratified, the dataset is split in a way where the class distribution of the train dataset is roughly equal to the test dataset.

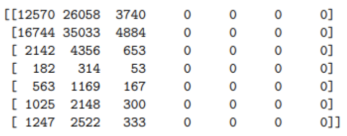
Concepts from class

One technique taught in class that was implemented in this project was normalization. With the tree cover dataset, the data varies from measurements in meters to boolean values. By normalizing the dataset you ensure that the model takes into account the values in reference to other values in the same feature. Rather than in reference to the other features.



```
             precision    recall  f1-score   support

          1       0.36      0.30      0.33     42368
          2       0.49      0.62      0.55     56661
          3       0.06      0.09      0.08      7151
          4       0.00      0.00      0.00       549
          5       0.00      0.00      0.00      1899
          6       0.00      0.00      0.00      3473
          7       0.00      0.00      0.00      4102

avg / total       0.38      0.42      0.39    116203
```

Fig. 9. classification report



```
[[12570 26058  3740     0     0     0     0]
 [16744 35033  4884     0     0     0     0]
 [ 2142  4356   653     0     0     0     0]
 [  182   314    53     0     0     0     0]
 [  563  1169   167     0     0     0     0]
 [ 1025  2148   300     0     0     0     0]
 [ 1247  2522   333     0     0     0     0]]
```

Fig. 10. confusion matrix

Analyzing Results

Regardless of normalization, feature selection, number of epochs, or hidden layer shape, the model consistently performed between 39% and 42% regardless of stratification. There seems to not be enough of a distinction between each class of trees because the model at its best performance appears to just guess randomly based of the expected distribution.

More Time

If I had more time I would try to look into whether or not there are other datasets for different areas which have similar data. I would train a model on the other dataset to see if the model has improved results, because I think that the dataset is the real problem. That is why there is no variation based on model changes. Once I have improved the data, I would run a optimization algorithm which would vary hidden layer shape, steps, and epochs. Run several tests for each instance, average the results, then record the best performing values.