

Department of Computer Engineering

INF508: Machine Learning
Project Report

Human Body Posture Classification Analysis
with
Different Algorithms and Sensor Combinations

Kemal Mert Makinacı

Galatasaray University

Fall 2017

1. Introduction:

The aim behind this project is, to learn machine learning fundamentals on a dataset, understand different data extraction and analysis approaches.

Parallel to increasing technology development, wearable device usage increases in daily life. Almost every people carries a smartphone which has gyroscope, accelerometer and many other sensor, high percentage of people wear smart watches which have almost same sensor with smart phones even more complicated ones. Moreover activity and sport related consumer electronic industry grows and launches new gadgets into market regularly. Because of this promising technology, the dataset used contains wearable accelerometer sample is used. It is reached from UCI Machine Learning repository. Dataset belongs to research “Wearable computing: Accelerometers’ Data Classification of Body Postures and Movements”¹. Dataset has 165632 number of instances which have 18 attributes. Data in instances consist of integer and real number values and 3 non-integer values. Non-integer values are converted to integer values in data preprocessing section. Subject to this conditions, dataset applicable for multi-class classification.

Python 3.6 programming language is used to extract, manipulate and analyze. In python coding scikit-learn tool used with help of pandas, scipy, numpy libraries. In project folder different modules are created to use as a cascaded machine learning library. More detailed information about python modules’ usage can be found in “readme.txt” file in project folder. Seven different algorithm used to analyze and compare results from algorithm output. Top three of them which have higher than 0.94% of accuracy, examined in more detail.

2. Methodology

The source research considers 5 activity classes which are sitting, sitting down, standing, standing up, and walking. The accelerometers were respectively positioned in the waist (1), left thigh (2), right ankle (3), and right arm (4), it is illustrated in Fig.1. Then accelerometers give raw data related with the sensor’s x, y, z plane position such as x1,y1,z1. 12 attributes derived from a time window of 150ms.

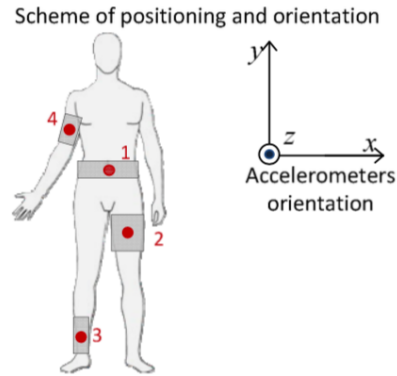


Fig.1 Sensor positioning on body.

Reference research claims that they use a classifier that uses a committee AdaBoost that combines ten Decision Trees, and observed classifier accuracy 99.4%. On this project common in use algorithms such as Logistic Regression, Gaussian Naïve Bayes, Decision Tree, Random Forest, K-Neighbors and Support Vector Machine classifiers tested and analyzed. Algorithms imported from python scikit-learn tool.

To analyze data firstly, program read in its own memory then identifies data section and label axis. The dataset information output is shown in Fig.2. Also data histogram is plotted in Fig.3.

```

Console  PyUnit
<terminated> data_visualization.py [usr/local/bin/python3.6]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165632 entries, 0 to 165631
Data columns (total 19 columns):
 .user                165632 non-null object
 gender              165632 non-null object
 age                165632 non-null int64
 how_tall_in_meters  165632 non-null float64
 weight             165632 non-null int64
 body_mass_index     165632 non-null float64
 x1                 165632 non-null int64
 y1                 165632 non-null int64
 z1                 165632 non-null int64
 x2                 165632 non-null int64
 y2                 165632 non-null int64
 z2                 165632 non-null int64
 x3                 165632 non-null int64
 y3                 165632 non-null int64
 z3                 165632 non-null int64
 x4                 165632 non-null int64
 y4                 165632 non-null int64
 z4                 165632 non-null int64
 class              165632 non-null object
 dtypes: float64(2), int64(14), object(3)
 memory usage: 24.0+ MB

Dataset Shape
(165632, 19)
Class Labels
['sitting' 'sittingdown' 'standing' 'standingup' 'walking']

```

Fig.2 Python Console Dataset Information Output

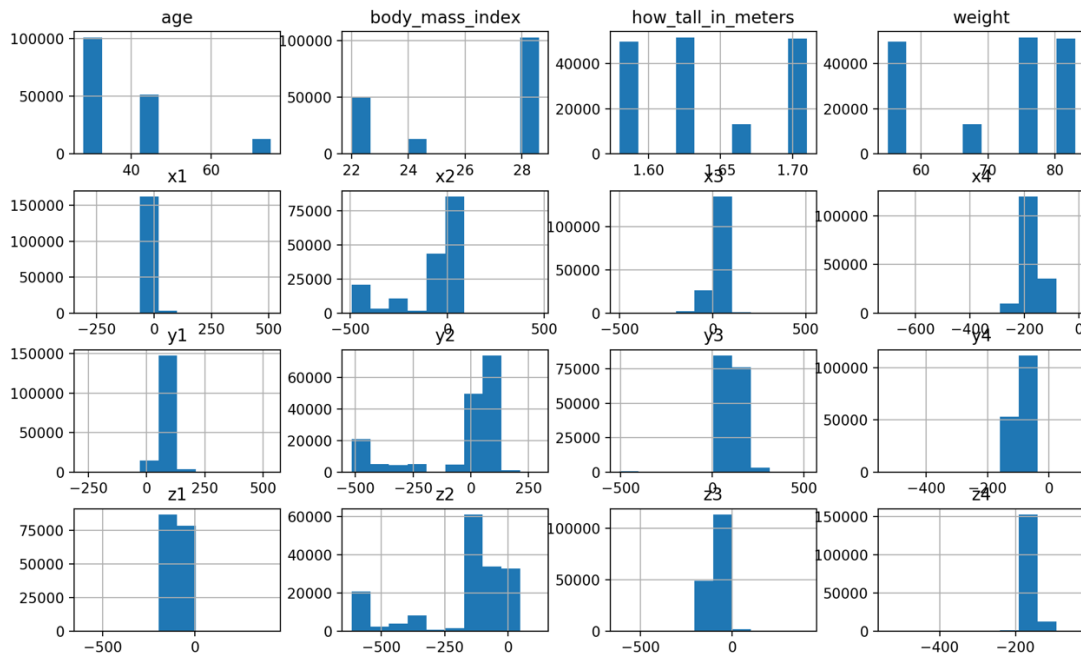


Fig.3 Data Histogram

Real values of sensors and human body feature can be shown until this point of program. However, machine learning algorithm does not need real and measured values all time, and also sometimes they are not capable to understand them. So that after the dataset integration, preprocessing applying to it, to put in shape which allowed in classification algorithms. In python pandas and numpy libraries make this operation easy. Label encoding, fit transform and standart scaler functions are applied. Label encoding is performed by scikit learn's preprocessing library LabelEncoder function (`sklearn.preprocessing.LabelEncoder`). It encodes label with value between 0 and `n_classes-1`. Standar scaling is done due to standardize features by removing the mean and scaling to unit variance. To apply this on this project `sklearn.preprocessing.StandardScale` function is called.

To decide which algorithm fits with our dataset, seven algorithm tested and analyzed. Brief explanations about algorithm are;

Decision Trees (DT): Subset of supervised learning algorithms family. With DT can solve both regression and classification problems. DT predicts class or value by learning decision rules inferred

from prior data. It algorithm; places the best attribute of dataset at the root of the tree, splits the training set into subsets and repeats these steps until the leaf node in all the branches of tree.

Random Forest: This algorithm uses decision tree classifier but the process of finding the root node and splitting the feature node will run randomly instead of using decision rule belongs the information gain or gain index in DT. Random Forest can handle missing values an categorical values.. Also if there enough trees in forest, the classifier will not overfit the model. ^[4]

Gaussian Naïve Bayes: This algorithm for binary and multi class classification problems. The technique is easiest to understand when described using binary or categorical input values. Calculation of probabilities for each hypothesis are simplified to make their calculations tractable. Rather than attempting to calculate the values of the each attribute value $P(d1, d2, d3|h)$, they are assumed to be conditionally independent given the target value and calculated as $P(d1|h) * P(d2|H)$ and so on. Although this is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold. ^[5]

K-Neighbors: This is a simple algorithm that stores all available cases and classifies new cases based on similarity measure. It is widely used in statistics and pattern recognition areas. A case is classified by a majority vote of its neighbors with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. This classifier is also non parametric and instance-based algorithm. ^[6]

Logistic Regression: Logistic regression is a statistics method and is used to binary classification problems. It has logistic function. Logistic regression assumes no error in the output variable(y), because of this data should be preprocessed for this algorithm. Preprocess should include removing correlated inputs, outliers and misclassified instances and correlated inputs. ^[7]

Support Vector Machine: SVM uses a flexible representation of the class boundaries to solve classification problems. It implements automatic complexity control to reduce overfitting. It is easy to use and often has good generalization performance. ^[8]

3. Evaluation

Evaluation process start with separating dataset in train and test subsets. As known overfitting occurs as a common problem in machine learning projects. Overfit means, model fit too much to training data. Overfit affect the predictability of model on test when model is too complex and causes accuracy problem. In contrast to overfitting, model can be underfitted. This means that model does not fit the training data and therefore misses the trends in the data. So that model cannot generalized to new data. Skitlearn has a model_selection library for model selection featured functions and train_test_split function performs splitting. This functions has basic splitting function. More complicated and reliable splitting performs with model_selection. K-fold function. K-fold function is used as illustrated in Fig.4. Cross-validation involves partitioning a sample of data into complementary subsets, analyzing perform in one subset which called train set, and validating performs on other set which called validation set or test set. Variability is a case, so to reduce it multiple rounds of cross-validation may perform using different subset assigning then it is called k-fold. Each validation of subset result averaged and final score predicted for that model.

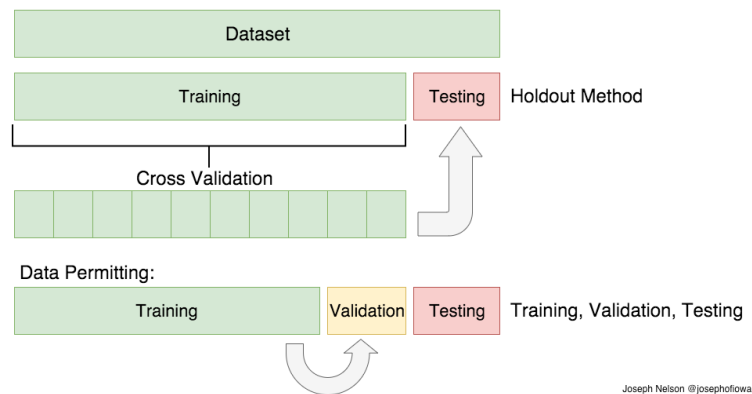


Fig.4 K-Fold Cross Validation Scheme

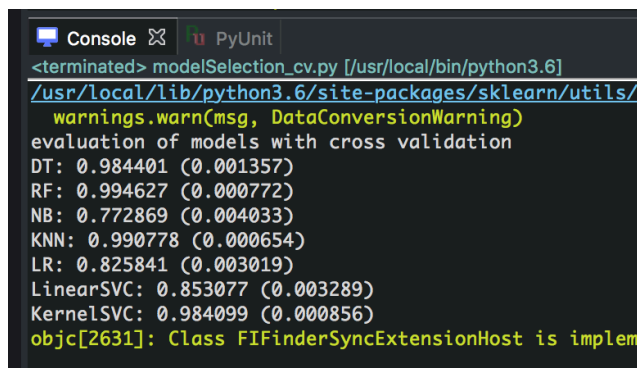
A. Model Selection without Cross Validation

This model selection procedure performs simple train test dataset splitting , then performs seven algorithms with parameters shown below:

- DecisionTreeClassifier(criterion = 'entropy')
- RandomForestClassifier(n_estimators=100, criterion= 'entropy')
- GaussianNB()
- KNeighborsClassifier(n_neighbors = 5)

- LogisticRegression()
- SVC(kernel= 'linear')
- SVC(kernel='rbf')

Generally accuracy is the first parameter to determine model efficiency and fit. Fig.5 shows the seven algorithms' accuracy score which is get by accuracy_score(Y_test, Y_prediction) function of scikitlearn. More detailed output can be shown on appendix II. Algorithms accuracy scores are satisfied generally especially Decision Tree, Random Forest , KNN and Kernel SVC algorithms give best results on dataset. These accuracy score is close to reference research and top rated accuracy scores commented at Appendix I.



```
<terminated> modelSelection_cv.py [/usr/local/bin/python3.6]
/usr/local/lib/python3.6/site-packages/sklearn/utils/
warnings.warn(msg, DataConversionWarning)
evaluation of models with cross validation
DT: 0.984401 (0.001357)
RF: 0.994627 (0.000772)
NB: 0.772869 (0.004033)
KNN: 0.990778 (0.000654)
LR: 0.825841 (0.003019)
LinearSVC: 0.853077 (0.003289)
KernelSVC: 0.984099 (0.000856)
objc[2631]: Class FIFinderSyncExtensionHost is implem
```

Fig.5 Model Selection without Cross Validation Output

Although accuracy score does not verify the algorithm performance by itself, for this example precision, recall, f1-score and support values are coherent with accuracy score. So that algorithm performance ranking can be done by accuracy score. Result of experiment state that Random Forest and KNN are top two algorithm for this dataset, Decision Tree and KernelSVC are the following algorithms which have still reasonably high accuracy scores.

B. Model Selection with K-Fold Cross Validation

After the first model selection experiments due to high score, a bit more deep examination needed. To detect any overfitting case k-fold cross validation applied on same algorithms and algorithm variables. In this experiment k selected as 10, it is also default value for many k-fold applies. The results given 1% more worse result on accuracy scores, but this result is gives us prior applied algorithms consistent on this dataset and there is no significant overfitting or other anomalies. Algorithm accuracy scores can be shown on Fig.6 graphically.

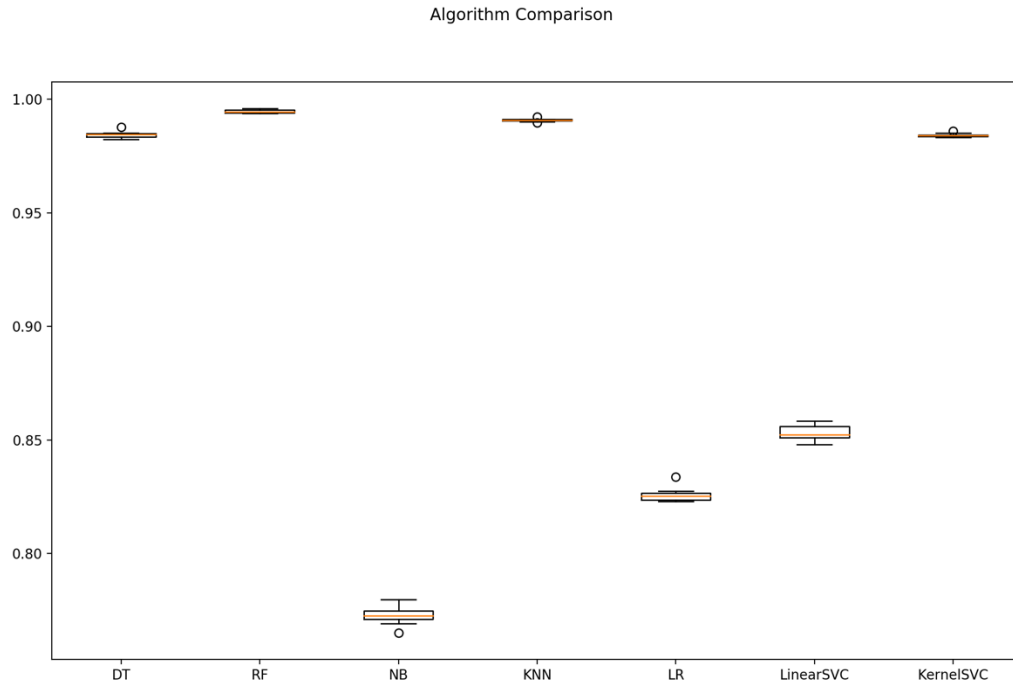


Fig.6 Algorithm Comparison with K-Fold Cross Validation

C. Sensor Usage Effectivity Analysis

As said before activity recognition gadget developed and released to market in fast. When a product is released then its competitors occurs to make it more reliable and cheaper. With this vision this dataset can be examined with less sensor and unless personal data. These examinations performed with top four algorithm previously chosen.

In a logic way activity recognition can be detected with movement, that movement will be more frequent on legs and arms on human body. So sensor data gathered from waist firstly removed. Then results observed as;

'DT': 0.98318592085006185, 'RF': 0.99375132067497807,
'KNN': 0.98460470311226489, 'KernelSVC': 0.97029613306366413

These result says that waist sensor does not give significant benefit.

Secondly, thigh sensor data removed. Thigh has more movement than the waist, on the other hand it has less movement than arm or leg ankle.

'DT': 0.97980499290608869, 'RF': 0.99266459383584382,
'KNN': 0.98714039907024487, 'KernelSVC': 0.98083134603193767

The last sensor removed from arm ankle which has most movement on body.

'DT': 0.97566939354605009, 'RF': 0.99015908473450664,
'KNN': 0.98167657801793096, 'KernelSVC': 0.96766987653575631

Another examination performed by removing personal data but with all sensor. The result again does not change significantly.

'DT': 0.9819482597277146, 'RF': 0.99462673951761404,
'KNN': 0.98750264134995624, 'KernelSVC': 0.97754097865789236

Finally, It can be state that on this dataset removing some data from dataset does not affect the overall result significantly.

4. Conclusion and Future Work

In conclusion Decision Tree, Random Forest, K-Neighbor, and Support Vector Machine with kernel parameter algorithms are good model for this dataset. However, this models fit well on this dataset and k-fold cross validation applied, more advanced overfitting method may be investigated fort his application. To prove this dataset and algorithm couples, re-gathered data may be used.

Beside the technical context, as an embedded software engineer I am working with mostly sensor data and their manipulation. Due to this project I have learned a way to analyze huge raw data to interpretation, and I have owned a sensor data mining library. This project may not be good example who is working on machine learning or data science specialty but I think it gives an initial coverage to new comers like me.

5. References

1. Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidui, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012.
2. Python Machine Learning, Sebastian Raschka, 2015, ISBN 978-1-78355-513-0
3. Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers

4. <https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674>

5. <https://machinelearningmastery.com/naive-bayes-for-machine-learning/>

6. http://www.saedsayad.com/k_nearest_neighbors.htm

7. <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>

8. <https://www.cs.cmu.edu/~ggordon/SVMs/new-svms-and-kernels.pdf>

Appendix I.

A list of metadata drawn from the most recent publications (2012 and 2011) is shown on Table 1.

Table 1. HAR based on accelerometers' data from 2012 and 2011 (IEEE database)

Research	# of sensors	Accelerometers' position	Solution	# of users	Learning mode	Test mode	Correct (%)
Liu et. al. (2012) [6]	1	hip, wrist (no info about orientation)	SVM	50	Super-vised	leave-one-out	88.1
Yuting et al. (2011) [7]	3	chest and both thighs (no info about orientation)	Threshold-based	10	--	--	98.6
Sazonov et al. (2011) [8]	1	foot	SVM	9	Super-vised	4-fold cross validation	98.1
Reiss & Stricker (2011) [9]	3	lower arm, chest and foot	Boosted Deci-sion Tree	8	Super-vised	8-fold cross validation	90.7
Min et al., (2011) [10]	9	torso, arms and legs	Threshold-based	3	--	Comparison with k-means	96.6
Maekawa & Watanabe (2011) [11]	4	wrists of both hands, waist, and right thigh	HMM	40	Unsu-pervised	leave-one-out	98.4
Martin et al. (2011) [12]	2	hip, foot and chest	Threshold-based	5	--	--	89.4
Lei et al. (2011) [3]	4	waist, chest, thigh, and side of the body	Naive Bayes	8	Super-vised	Several, w/ no cross validation	97.7
Alvarez et al. (2011) [13]	1	centered in the back of the person	Genetic fuzzy finite state machine	1	Super-vised	leave-one-out	98.9
Jun-ki & Sung-Bae (2011) [14]	5	forehead, both arms, and both wrists	Naive Bayes and SVM	3	Super-vised	leave-one-out	99.4
Ioana-Iuliana & Rodica-Elena (2011) [15]	2	right part of the hip, lower part of the right leg	Neural Networks	4	Super-vised	66% training vs 33% test	99.6
Gjoreski et al. (2011) [2]	4	chest, waist, ankle and thigh	Naive Bayes, SVM, C4.5, Random Forest	11	Super-vised	Leave-one-person-out	90.0
Feng, Meiling, and Nan (2011) [16]	1	Waist	Threshold-based	20	--	--	94.1
Czabke, Marsch, and Lueth (2011) [17]	1	Trousers' Pocket	Threshold-based	10	--	--	90.0
Chernbumroong, et al. (2011) [18]	1	Non-dominant wrist (watch)	C4.5 and Neural Networks	7	Super-vised	5-fold cross-vali-dation	94.1
Bayati & Chavarriaga (2011) [19]	--	Simulations instead of real accelerometers	Expectation Maximization	--	Unsuper-vised	Not mentioned	86.9
Atallah et al (2011) [20]	7	ear, chest, arm, wrist, waist, knee, and ankle	Feature Selection algorithms*	11	Super-vised	Not applied	--
Andreu et al. (2011) [21]	1	Not mentioned	fuzzy rule-based	--	Online learning	--	71.4

* This work is about sensor positioning and feature extraction

Appendix II.

Decision Tree

confusion matrix

```
[[10082    1    0    10    0]
 [    7  2264    10    57    24]
 [    0    8  9241    21    55]
 [    8    57    23  2405    36]
 [    3    36    99    27  8653]]
```

accuracy score

0.985449935098

classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10093
1	0.96	0.96	0.96	2362
2	0.99	0.99	0.99	9325
3	0.95	0.95	0.95	2529
4	0.99	0.98	0.98	8818
avg / total	0.99	0.99	0.99	33127

#####

Random Forest

confusion matrix

```
[[10090    0    0    3    0]
 [    1  2340    5    7    9]
 [    0    0  9297    4   24]
 [    2    25    19  2471   12]
 [    0    5    13    2  8798]]
```

accuracy score

0.99604552178

classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10093
1	0.99	0.99	0.99	2362
2	1.00	1.00	1.00	9325
3	0.99	0.98	0.99	2529
4	0.99	1.00	1.00	8818
avg / total	1.00	1.00	1.00	33127

#####

Naive Bayes

confusion matrix

```
[[9234  93  706  56   4]
 [ 201 1359  630  31  141]
 [  62  103 8651  29  480]]
```

```
[ 232  512 1024  353  408]
[ 159  359 1772  563 5965]]
```

accuracy score
0.771636429499

```
classification report
      precision    recall  f1-score   support

     0       0.93      0.91      0.92     10093
     1       0.56      0.58      0.57      2362
     2       0.68      0.93      0.78      9325
     3       0.34      0.14      0.20      2529
     4       0.85      0.68      0.75      8818

 avg / total       0.77      0.77      0.76     33127
```

```
####      KNN      ####
confusion matrix
[[10082      0      0      9      2]
 [      0 2330     13     18     1]
 [      1      3  9298     14     9]
 [      5     35     38 2443     8]
 [      2     28    117     23  8648]]
```

accuracy score
0.990159084735

```
classification report
      precision    recall  f1-score   support

     0       1.00      1.00      1.00     10093
     1       0.97      0.99      0.98      2362
     2       0.98      1.00      0.99      9325
     3       0.97      0.97      0.97      2529
     4       1.00      0.98      0.99      8818

 avg / total       0.99      0.99      0.99     33127
```

```
#####      #####
Logistic Regression
confusion matrix
[[10057     14      2     20      0]
 [    170 1254     534    275    129]
 [      0     16  8010      8   1291]
 [    179    351     552   1202    245]
 [      6      89   1816    148   6759]]
```

accuracy score
0.823557822924

```
classification report
      precision    recall  f1-score   support

     0       0.97      1.00      0.98     10093
     1       0.73      0.53      0.61      2362
     2       0.73      0.86      0.79      9325
```

3	0.73	0.48	0.57	2529
4	0.80	0.77	0.78	8818
avg / total	0.82	0.82	0.82	33127

Linear SVC

confusion matrix

```
[[10055 12 0 26 0]
 [ 62 1833 176 222 69]
 [ 0 75 7867 11 1372]
 [ 70 262 436 1618 143]
 [ 7 428 1435 115 6833]]
```

accuracy score

0.851450478462

classification report

	precision	recall	f1-score	support
0	0.99	1.00	0.99	10093
1	0.70	0.78	0.74	2362
2	0.79	0.84	0.82	9325
3	0.81	0.64	0.72	2529
4	0.81	0.77	0.79	8818
avg / total	0.85	0.85	0.85	33127

Kernel SVC

confusion matrix

```
[[10078 0 0 13 2]
 [ 9 2300 23 18 12]
 [ 1 4 9272 18 30]
 [ 12 77 52 2380 8]
 [ 0 32 235 21 8530]]
```

accuracy score

0.982884052284

classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10093
1	0.95	0.97	0.96	2362
2	0.97	0.99	0.98	9325
3	0.97	0.94	0.96	2529
4	0.99	0.97	0.98	8818
avg / total	0.98	0.98	0.98	33127