Homework 1 - Machine Learning

A taste for music

Alexandra Dinisor, 342 C4

Modified FMA: A Dataset For Music Analysis: https://github.com/mdeff/fma

- Balanced dataset with 500 samples for train and 100 samples for test on each label
- Labels = genre_top: *Rock*, *Hip-Hop*, *Folk* & *Electronic*

1. KMeans clustering
   1.1. Baseline

Because the dataset is already labeled, K Means algorithm is not a particularly good choice.

However, Silhouette score can be used to draw Silhouette plots with **YellowBrick - a machine learning visualization** (https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam) for a different number of clusters and perform Silhouette analysis appropriately to find the most appropriate cluster between a certain range [2, 6]. (Fig. 1)

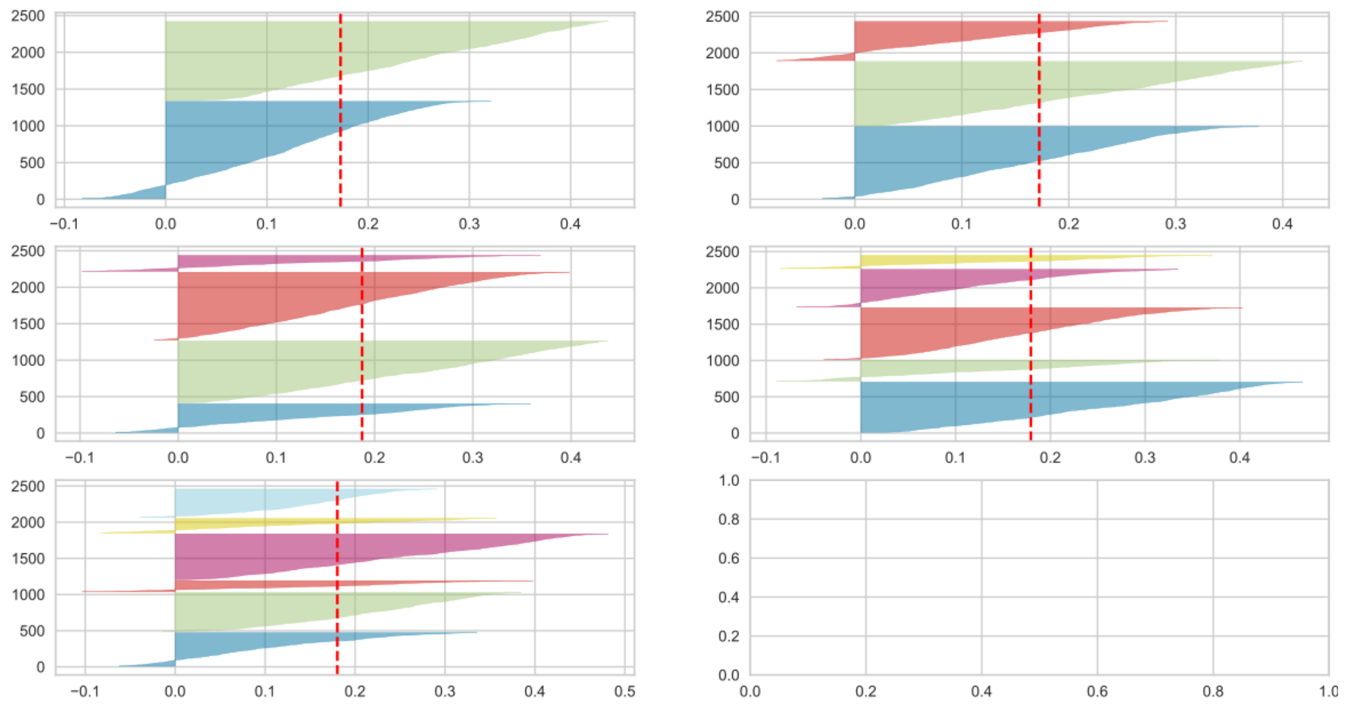


Fig. 1

The value of n_clusters as 5 and 6 looks to be suboptimal, due to the wide fluctuations in the size of the silhouette plots.

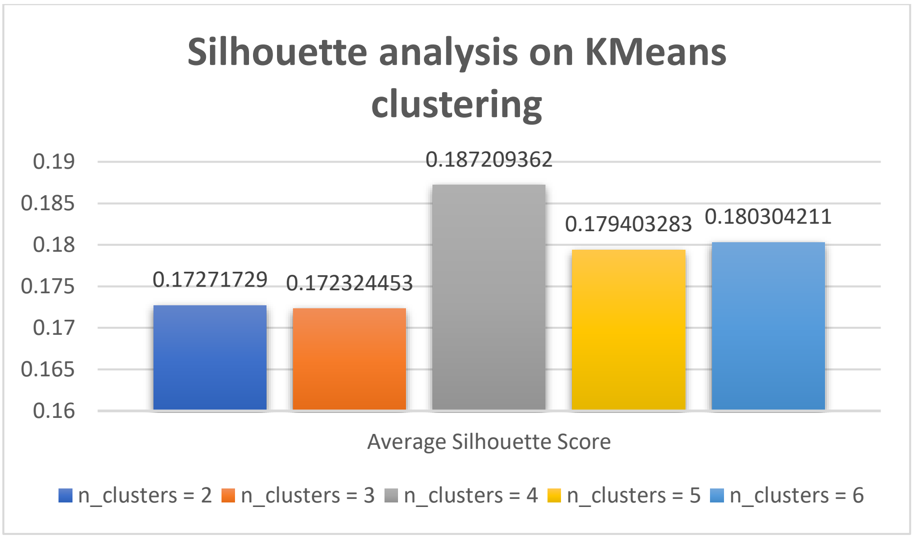


Fig. 2.

The average Silhoutte Score computed for each cluster shows that n_clusters = 4 (avg silhouette score = 0.1872093622274093) is the closest to the value 1, which means that the clusters are dense enough and well-separated than other clusters. (Fig. 2)

As a result, the optimal Kmeans solution has n_clusters = 4 with emphasize on the centroids (the black circles) in Fig. 3. with the minimum amount of features (only audio_features selected) - 8 features.

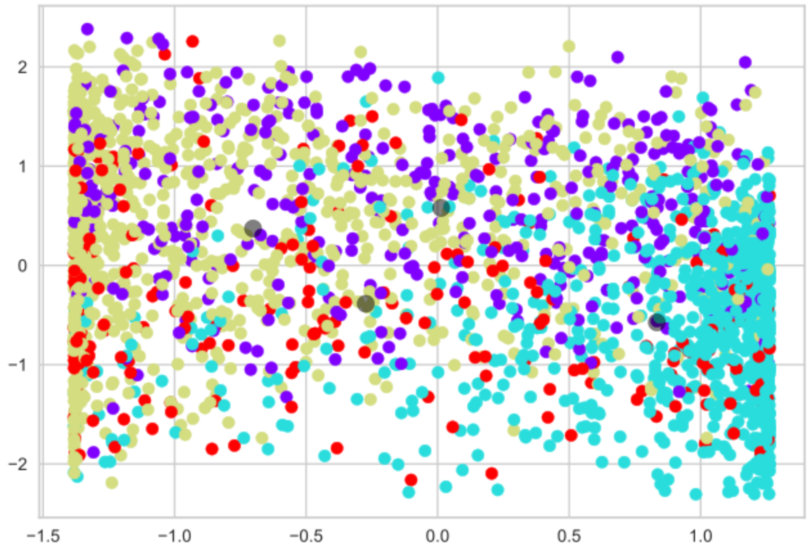


Fig. 3 Baseline Kmeans

1.2. Improved Kmeans model

I used feature selection to select the best k features with **sklearn.feature_selection**.SelectKBest from the whole features availables 237 in total (audio_features, social_eatures, temporal_features) and concluded an improved model with k =60 features (Fig. 4.)

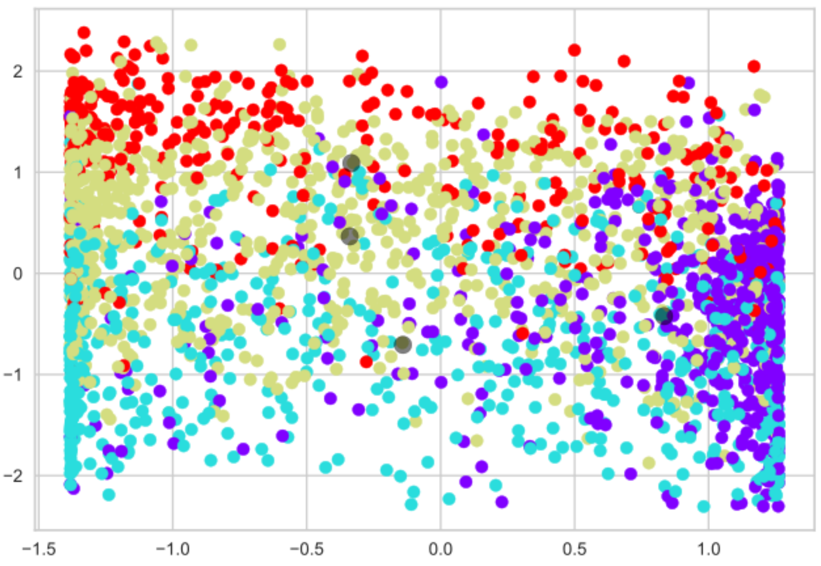


Fig. 4. Improved KMeans

Accuracy on KMeans clustering algorithm shows a 0.03 improvement measured with randIndex (Fig. 5.)

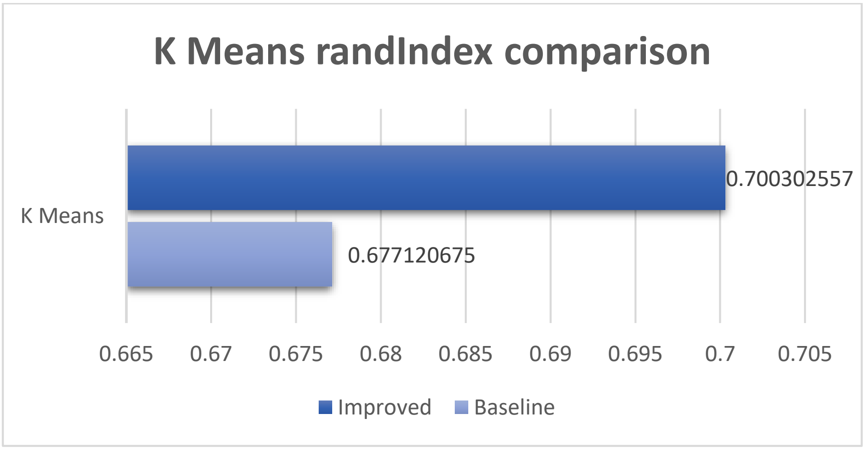


Fig. 5.

2. Random Forest
   2.1. Baseline

Radom Forest on baseline has only n_estimators=300 number of trees in the forest.

2.2. Impoved model

Improving the predictive power of the model, tuning the below listed hyperparameters using GridSearchCV helped.

parameters_rf = {'n_estimators': [400, 500, 600],

'max_features': ['auto'],

'criterion': ['entropy', 'gini'],

'max_depth': [5, 10]

}

And concluded on RandomForestClassifier(n_estimators=500, max_depth=10).

Furthermore, I added feature selection using 10 best features from 2 sets of features (audio_features, social_features).

Accuracy improvement of 0.4% (Fig. 6.)

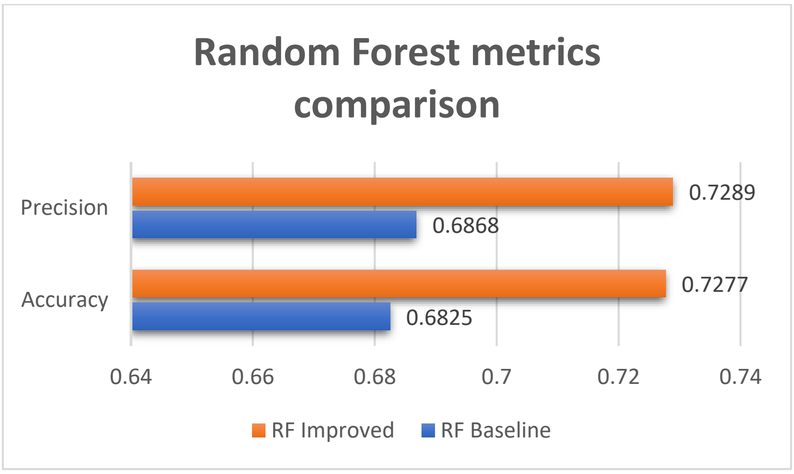


Fig. 6.

3. XGBoost
    3.1. Baseline

Baseline has only the minimum feature number from audio_features.

    3.2. Improved model

Hyperparameter tuning on XGBoost with GridSearchCV combined more params from the following list:

```
param_test = {
  'max_depth':range(3,10,2),
  'min_child_weight':range(1,6,2)
  }
```

And found: **XGBClassifier**(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1, importance_type='gain', interaction_constraints='', learning_rate=0.300000012, max_delta_step=0, max_depth=5, min_child_weight=3, missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=4, num_parallel_tree=1, objective='multi:softprob', random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None, subsample=1, tree_method='exact', validate_parameters=1, verbosity=None)

Another method I found helpful was feature_importance_ ploted in Fig.7. with features from audio_features and social_features and decided to remove 3 of them (closest to 0).

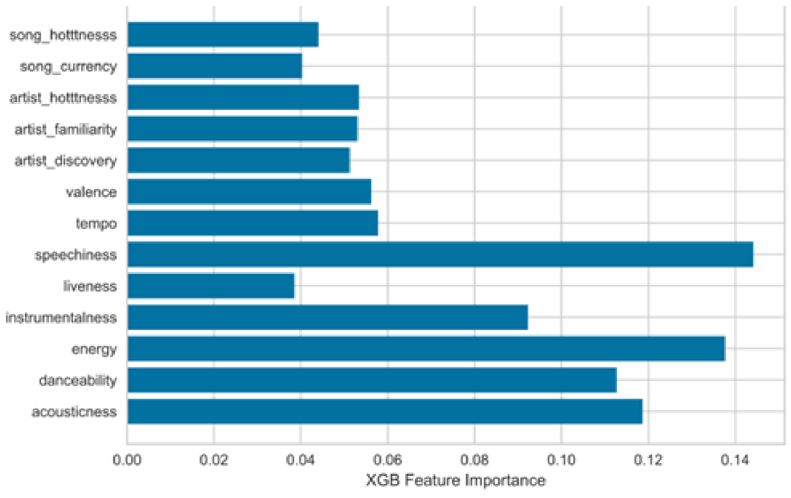


Fig. 7

In the end, precision and accuracy showed improvement of 11%. (Fig. 8.)

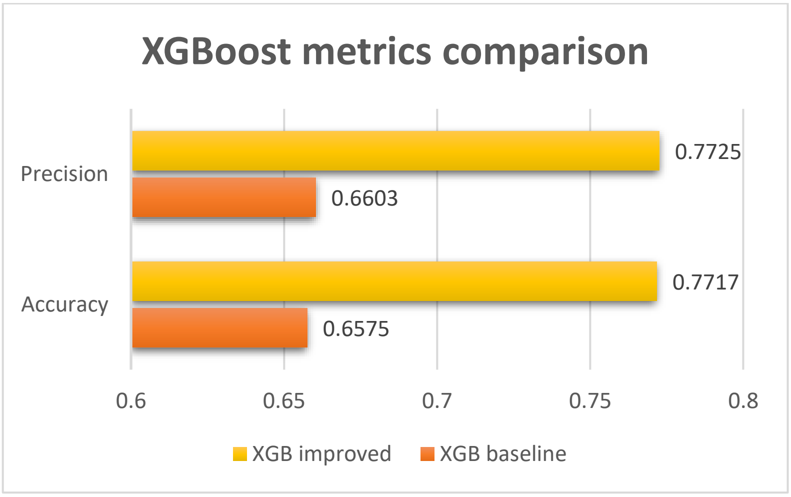


Fig. 8.

4. SVM

4.1. Baseline

SVM baseline uses a Radial basis function (rbf) kernel and audio_features.

4.2. Improved model

Tuning the hyperparameters with GridSearchCV, using Linear kernel and "one-versus-one" approach for multi-class classification turned out to be the best solution.

Playing aroung with the regularization parameter C, shows the best performance of SVM for linear kernel.

Following list for hyperparam tuning:

param_grid = {'C': [0.01, 0.1, 1, 10, 100],

'kernel': ['rbf','linear'],

'decision_function_shape':['ovo']}

An additional method of feature selection with 10 best features was useful.

Accuracy improvement of 39,2% and Precision improvement of 42,8%. (Fig. 9)

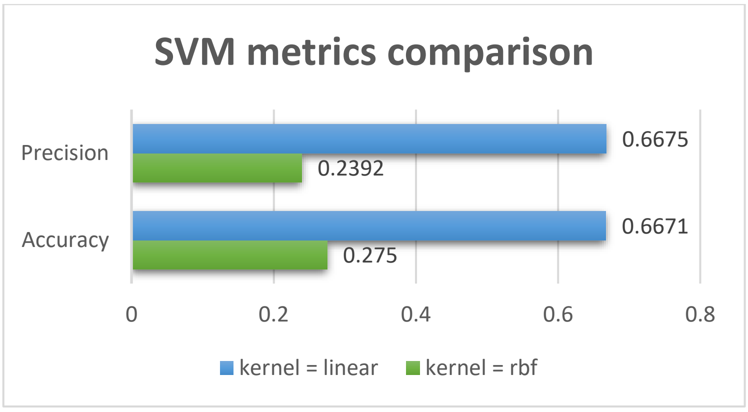


Fig. 9

5. Naive Bayes

5.1. Baseline

The most suitable Naive Bayes classifier for out multi-class classifier with balanced data is Gaussian Naive Bayes.

The other NB classifiers fail to match the model. For example, ComplementNB is suited for imbalanced datasets, BernoulliNB expects features as binary values.

From my experiment (Fig. 10) without using cross-validation on the baseline models, it is obvious that GaussianNB is the right one.

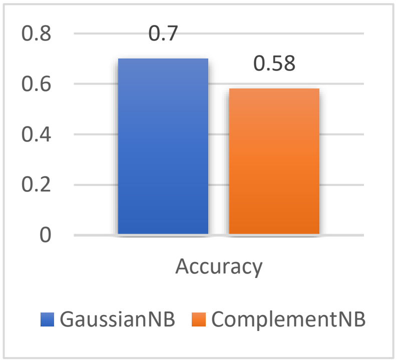
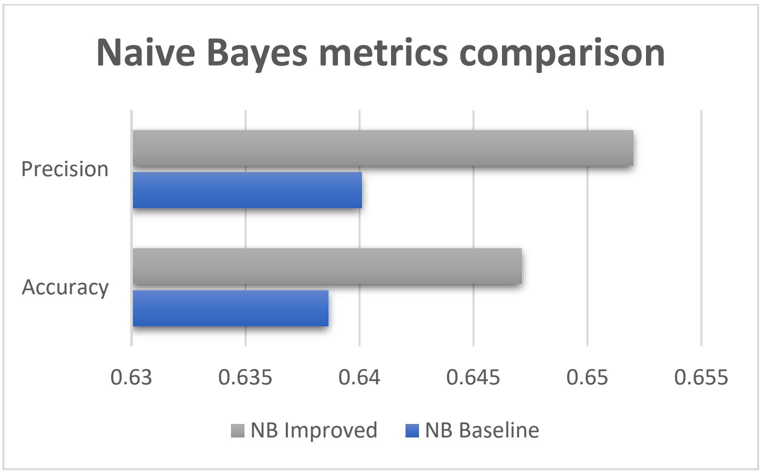


Fig. 10.

5.2. Model improved

A light improvement was only made with one hyperparameter found by GridSearchCV:

GaussianNB(var_smoothing=2.310129700083158e-06). - Fig. 11



6. Comparison between classifiers

After evaluating performance on all improved multi-class classifiers (plots listed below in Fig. 12 -14), the supervelised machine learning approach with XGBoost classifier performs best on this dataset. As shown in the experiment above, XGBoost is built by the idea of reducing the total error and shows better performance after removing irrelevant features from the model.

Kmeans also shows a surprising performance for a clustering algorithm when analysing randIndex with **sklearn.metrics**.rand_score.

Playing around with regularization parameters (e.g. parameter C for SVM), hyperparameters (e.g. n_estimators, min_sample_leaf for Random Forest) and feature selection (sklearn.feature_selection) have proved to show good improvement methods.

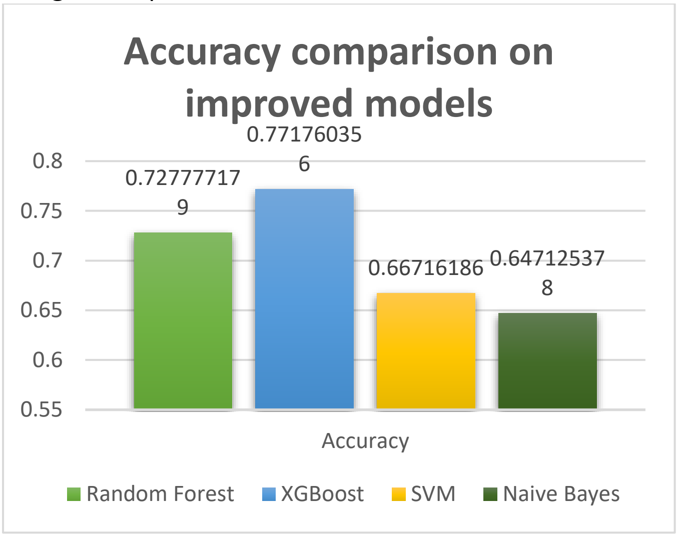


Fig. 12.

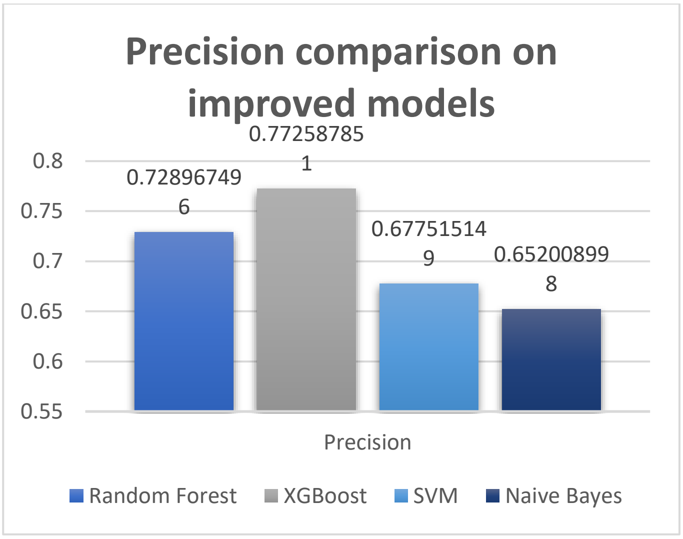


Fig. 13.

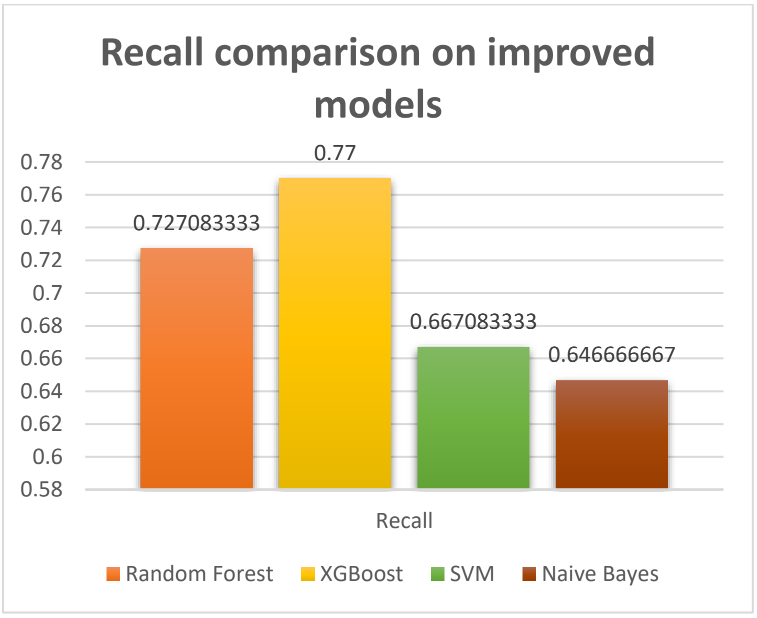


Fig. 14.