PROJECT 3

GROUP 12
Gao, Qing qg2175
Liu, Yuqiao yl4278
Wolansky, Ivan iaw2110
Yan, Xiyao xy2431
Zhu, Huizhe hz2657 (Presenter)

1. IDENTIFY THE TASK

TASK

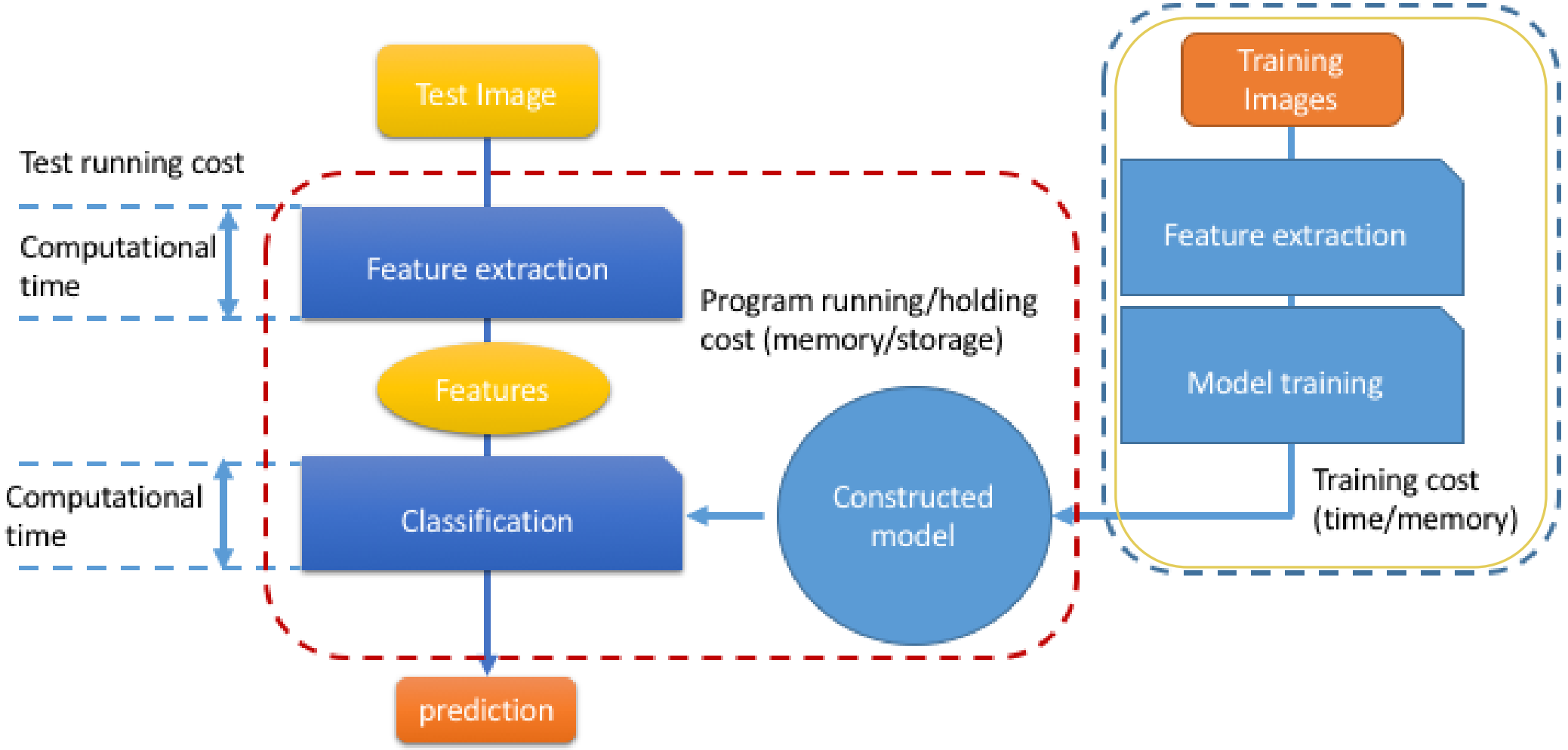
1. Task: Identify facial expression from 22 emotions -> classification problem
2. What we have

- 2500 images (training data set), info about 78 fiducial points for each image
- Build features: distances of fiducial points

3. What models

1) Linear based: Lasso, Ridge
2) Tree based: random forest
3) Mathematically: Support Vector Machine
4) Ensemble methods: Voting, Bagging, Gradient Boosting Machines

STRUCTURE


Test Image
Test running cost
Computational time
Feature extraction
Program running/holding cost (memory/storage)
Features
Computational time
Classification
Constructed model
prediction
Training Images
Feature extraction
Model training
Training cost (time/memory)

WORKFLOW

1. Feature extraction
2. Build baseline model
3. Build advanced models
4. Choose the best & build ensemble model

2. BUILD MODELS

FEATURE EXTRACTION

Feature extraction in Python

1. Compute 'Euclidean distance' between fiducial points – 3003 features (78*77/2 = 3003)
2. Extract features from fiducial points, get X_train (features)
3. Extract responses from 'label' file Y_train = train_label['emotion_idx']

File path: Spring2020-Project-group12/doc/Features Extraction --- R version.Rmd

Result:
Time for constructing features: 19.32 secs

BASELINE MODEL (GBM): 22.8%

1. **Train model**

- gbm.baseline <- train_gbm(train.df=dat_train, s=0.001, K=2, n=50)

2. **Predict on training and testing data**

- Pred_gbm_baseline <- test_gbm(gbm.fit=gbm.baseline, input.test=dat_test[, -6007], n=100)

3. **Get accuracy score**

- score = mean(dat_test$label == pred_gbm_baseline)

4. **Get estimated time training data**

- system.time(gbm.baseline <- train_gbm(train.df=dat_train, s=0.001, K=2, n=50))

Result:
Accuracy of GBM: 22.8%.
Time: 3.27 secs

VOTING CLASSIFIER: 52%

1. **Feature extraction**

3. **Train model:**

- estimators = [('svm', svm), ('ridge', ridge), ('logi', logistic)]
- voting = VotingClassifier(estimators=estimators, voting='hard') # majority vote
- voting.fit(x_train_pca, y_train)

4. **Predict on training and testing data**

- Test_pred = voting.predict(x_test_pca)
- Train_pred = voting.predict(x_train_pca)

5. **Get accuracy score**

- Train_score = accuracy_score(Y_train, train_pred)
- Test_score = accuracy_score(y_validation, test_pred)

6. **Get estimated time training data**

Result:
Accuracy: 52.32%
Average time: 2.57 secs

BAGGING CLASSIFIER (SVM + PCA): 51.6%

1. **Feature extraction**
2. **Set model parameters**

- Svm = SVC(kerner='linear', C = 0.0001)
- Svm_bag = BaggingClassifier(svm, n_estimators=500, n_jobs=-1, verbose=3)
- Pca = PCA(n_components=128)

3. **Apply pca**

- X_train_pca = pca.fit_transform(X_train)

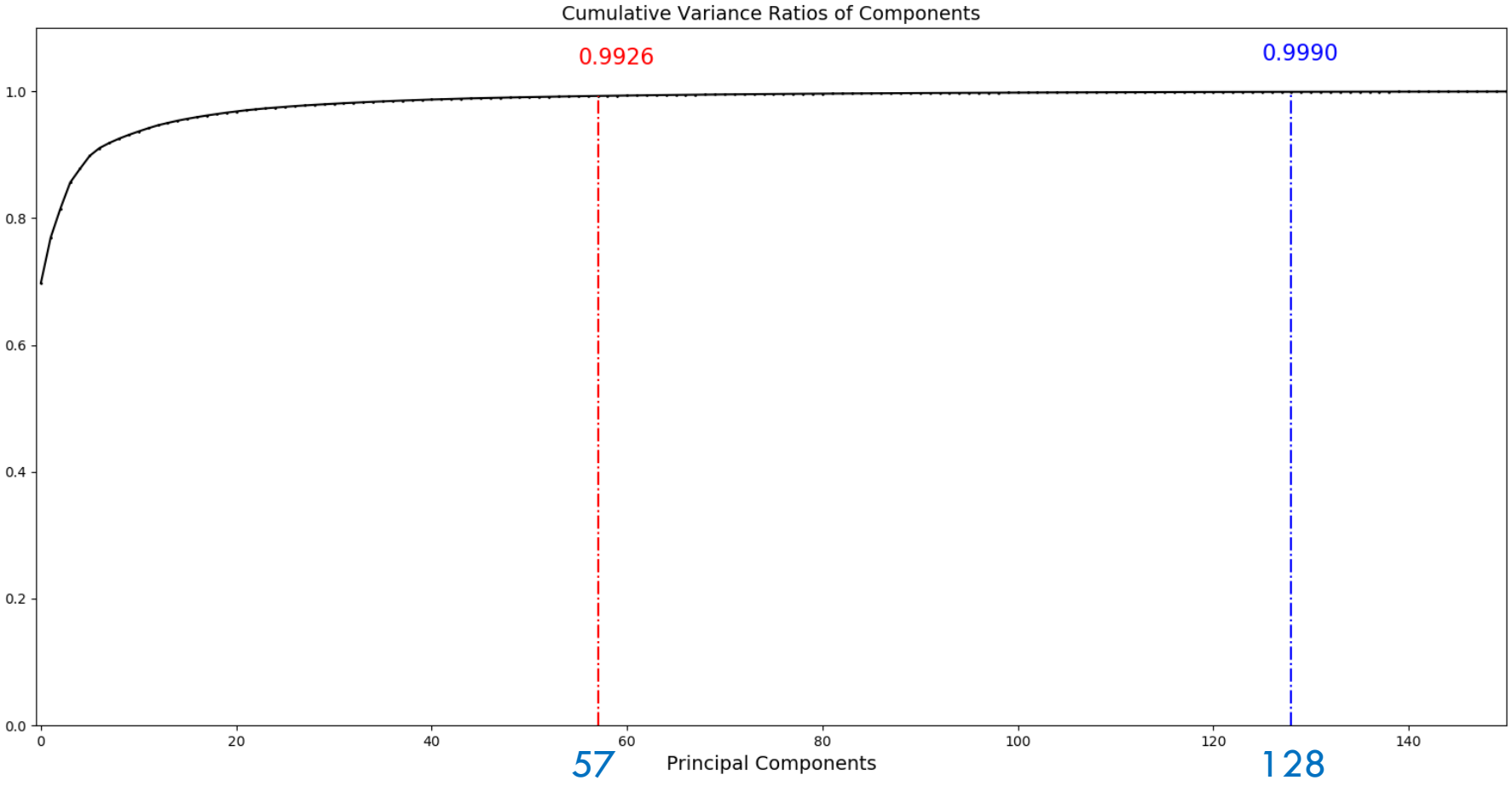
4. **Train model:**

- Svm_bag.fit(X_train_pca, y_train)

4. **Predict on training and testing data**
5. **Get accuracy score**
6. **Get estimated time**

Result:
Accuracy: 51.6%
Time: ~21.7 secs

PCA: DETERMINE NUMBER OF COMPONENTS



LASSO: 54%

Regularization method 1: set the less contributive variables to 0 (fast)

Steps

1. Feature extraction
2. Change the type data, convert x_train to matrix of predictors, required by glmnet(), convert y_train to factor
3. Train model with 5-folds cross validation, find the best lambda defines shrinkage amount

- cv.glmnet(x= X_train, y = y_train, family = 'multinomial', type.measure = 'class', nfolds = 5)

- Specify alpha = 1

Result:
Accuracy: 54.28 %
Time: 15.4 min train model, 11.5 secs to test

RIDGE: 49.6%

Regularization method 2: set the less contributive variables close to 0

Steps

1. Feature extraction
2. Change the type data, convert x_train to matrix of predictors, required by glmnet(), convert y_train to factor
3. Train model with 5-folds cross validation, find the best lambda

- cv.glmnet(x= X_train, y = y_train, family = 'multinomial', type.measure = 'class', nfolds = 5)

- Specify alpha = 0

Result:
Accuracy: 49.58%
Time: 32.9 min train model, 12.7 secs to test

SVM + PCA: 50.8%

1. **Feature extraction**

2. **train_pca, train SVM by the first 500 components**

- Svm(train_pca$x[,1:i], y = as.factor(data.train[,3004]))

3. **Select the first 57th components (performs best on the validation set)**

- Model_svm<-svm_list[[57]]

4. **Predict data and get accuracy**

Result:
Accuracy: 50.8%
Time: > 3 hours

3. SUMMARY

MODEL SUMMARY: LASSO WORKS BEST

1. Baseline Model: GBM: 22.8% (3.27 min)
2. Voting Classifier: SVM + Ridge + Logistic Regression: 52.32%, (2.57 secs)
3. SVM with Bagging method: 51.6% (~21.7 secs)
4. Lasso: 54.28% (15.4 min) to train model
5. Ridge: 49.58% (32.93 min) to train model
6. PCA+SVM: 57 PCs, 50.8% accuracy (>3 hours for training model)

IN THE END

To summarize:

- Baseline model: 22.8%
- Advanced models: 49.58% - 54.28%

➢ Lasso performs the best

Final thoughts

1. Why lasso works best? too many features (3003), by setting less contributive features to 0
2. Why ensemble method does not work best? Bias may exist, from the dataset etc.

THANK YOU