COSC 3337 : Data Science I

N. Rizk

College of Natural and Applied Sciences

Department of Computer Science

University of Houston

Feature Reduction (extraction) vs. Feature Selection

- Feature reduction
  - All original features are used
  - The transformed features are linear combinations of the original features
- Feature selection
  - Only a subset of the original features are selected
- Continuous versus discrete

Feature Selection vs Dimensionality Reduction

- Feature selection is simply selecting and excluding given features without changing them.

- Dimensionality reduction (Feature extraction) transforms features into a lower dimension.

Feature Selection

1. Remove features with missing values
2. Remove features with low variance
3. Remove highly correlated features
4. Univariate feature selection
5. Recursive feature elimination
6. Feature selection using SelectFromModel

Dimensionality Reduction

PCA

Baseline Model

```
# prepare for modeling
X_train_df = train.drop(['id', 'target'], axis=1)
y_train = train['target']
# scaling data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train_df)
lr = LogisticRegression(solver='liblinear')
lr_scores = cross_val_score(lr,
                            X_train,
                            y_train,
                            cv=5,
                            scoring='roc_auc')
print('LR Scores: ', lr_scores)
LR Scores:  [0.80729167 0.71875    0.734375   0.80034722 0.66319444]
```

The model is overfitting from the variation in cross validation scores. We can attempt to improve these scores through feature selection.

1-Remove features with missing values

```
# check missing values
train.isnull().any().any()
False
```

2-Remove features with low variance

```
from sklearn import feature_selection

sel = feature_selection.VarianceThreshold()
train_variance = sel.fit_transform(train)
train_variance.shape
(250, 302)
```

3-Remove highly correlated features

```
# find correlations to target
corr_matrix = train.corr().abs()
print(corr_matrix['target'].sort_values(ascending=False).head(1
0))
target    1.000000
33        0.373608
65        0.293846
217        0.207215
117        0.197496
91        0.192536
24        0.173096
295        0.170501
73        0.167557
183        0.164146
```

the features that are most highly correlated with our target variable. Feature 33 has the highest correlation to the target, but with a correlation value of only 0.37, it is only weakly correlated.

Drop features with a correlation value >0.5

```
# Find index of feature columns with high correlation
to_drop = [column for column in matrix.columns if any(matrix[column] > 0.50)]
print('Columns to drop: ' , (len(to_drop)))
Columns to drop: 0
```

4-Univariate feature selection

Univariate feature selection works by selecting the best features based on univariate statistical tests.

```
from sklearn.feature_selection import SelectKBest, f_classif
# feature extraction
k_best = SelectKBest(score_func=f_classif, k=100)
# fit on train set
fit = k_best.fit(X_train, y_train)
# transform train set
univariate_features = fit.transform(X_train)
```

5-Recursive feature elimination

Recursive feature selection works by eliminating the least important features. It continues recursively until the specified number of features is reached

```
from sklearn.feature_selection import RFE
# feature extraction
rfe = RFE(rfc, n_features_to_select=100)
# fit on train set
fit = rfe.fit(X_train, y_train)
# transform train set
recursive_features = fit.transform(X_train)
```

6-Feature Selection using SelectFromModel

Like recursive feature selection, sklearn's SelectFromModel is used with any estimator that has a coef_ or feature_importances_ attribute. It removes features with values below a set threshold.

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
# define model
rfc = RandomForestClassifier(n_estimators=100)
# feature extraction
select_model = feature_selection.SelectFromModel(rfc)
# fit on train set
fit = select_model.fit(X_train, y_train)
# transform train set
model_features = fit.transform(X_train)
```

PCA (Principle Component Analysis) is a dimensionality reduction technique that projects the data into a lower dimensional space.

PCA can be useful in many situations, but especially in cases with excessive multicollinearity or explanation of predictors is not a priority

```
from sklearn.decomposition import PCA
# pca - keep 90% of variance
pca = PCA(0.90)
principal_components = pca.fit_transform(X_train)
principal_df = pd.DataFrame(data = principal_components)
print(principal_df.shape)
(250, 139)
```

we are left with 139 features that explain 90% of the variance in our data.