UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Finding Donors for CharityML

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Great work on this project implementation!! Excellent and impressive work on data preprocessing, model selection, reasoning behind each selection, discussing pros and cons, correctly training and evaluating results on each sample of 1, 10 and 100%.

You have perfectly justified the selection of best model and further fine tuned that using GridSearchCV approach to find best parameter estimates. Using those estimates you evaluated final model score and compared against prior score to check comparative improvement.

At last you did great on feature selection using feature importance and evaluating how much contribution key features have on model outcome. This is good machine learning case study which requires in depth analysis and knowledge on ML techniques. You have gained good expertise on this approach. Congrats for finishing this project and all the best for other projects in this Nanodegree!!

for finishing this project and all the best for other projects in this Nanodegree!!

Exploring the Data

Student's implementation correctly calculates the following:

- Number of records
- Number of individuals with income >$50,000
- Number of individuals with income <=$50,000
- Percentage of individuals with income > $50,000

Well done evaluating these numbers correctly!!

```
Total number of records: 45222
Individuals making more than $50,000: 11208
Individuals making at most $50,000: 34014
Percentage of individuals making more than $50,000: 24.78439697492371%
```

Preparing the Data

Student correctly implements one-hot encoding for the feature and income data.

Good work on one hot encoding!! Its correctly displaying 103 total features after one-hot encoding.

```
103 total features after one-hot encoding.
```

Learn more about different ways to encode variables at this link.
https://machinelearningmastery.com/how-to-prepare-categorical-data-for-deep-learning-in-python/

Evaluating Model Performance

Student correctly calculates the benchmark score of the naive predictor for both accuracy and F1 scores.

Well done evaluating Accuracy and FScore for naive predictor.

```
Naive Predictor: [Accuracy score: 0.2478, F-score: 0.2917]
```

Benchmark process can be simple model algorithm or naive solution approach. Idea is to establish initial results upon which further improvements can be made in final model solution to assess the relative improvements.

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to be explored.

Please list all the references you use while listing out your pros and cons.

Good work discussing pros and cons, area of application and why consider consider these algorithm for this case study.

Here's further reference on how to choose right estimator.
https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Student successfully implements a pipeline in code that will train and predict on the supervised learning algorithm given.

Good work on train and predict pipeline implementation.
Well done sampling training data and evaluating accuracy_score and fbeta_score.

```
    # Compute accuracy on the first 300 training samples which is y_train[:300]
    results['acc_train'] = accuracy_score(y_train[:300], predictions_train)

    # Compute accuracy on test set using accuracy_score()
    results['acc_test'] = accuracy_score(y_test, predictions_test)

    # Compute F-score on the the first 300 training samples using fbeta_score()
    results['f_train'] = fbeta_score(y_train[:300], predictions_train, beta=.5)

    # Compute F-score on the test set which is y_test
    results['f_test'] = fbeta_score(y_test, predictions_test, beta=.5)
```

Student correctly implements three supervised learning models and produces a performance visualization.

Well done evaluating scores on each sample size for training and test datasets and setting the random state in classifier to make model reproducible.

```
# Initialize the three models
clf_A = SVC()
clf_B = DecisionTreeClassifier()
clf_C = GaussianNB()

# Calculate the number of samples for 1%, 10%, and 100% of the training data
# HINT: samples_100 is the entire training set i.e. len(y_train)
# HINT: samples_10 is 10% of samples_100 (ensure to set the count of the values to be `int` and not `float`)
# HINT: samples_1 is 1% of samples_100 (ensure to set the count of the values to be `int` and not `float`)
samples_100 = int(len(y_train))
samples_10 = int(len(y_train)*.1)
samples_1 = int(len(y_train)*.01)

# Collect results on the learners
results = {}
for clf in [clf_A, clf_B, clf_C]:
    clf_name = clf.__class__.__name__
    results[clf_name] = {}
    for i, samples in enumerate([samples_1, samples_10, samples_100]):
        results[clf_name][i] = \
        train_predict(clf, samples, X_train, y_train, X_test, y_test)

# Run metrics visualization for the three supervised learning models chosen
vs.evaluate(results, accuracy, fscore)
```

```
SVC trained on 361 samples.
SVC trained on 3617 samples.
SVC trained on 36177 samples.
DecisionTreeClassifier trained on 361 samples.
DecisionTreeClassifier trained on 3617 samples.
```

Improving Results

Justification is provided for which model appears to be the best to use given computational cost, model performance, and the characteristics of the data.

Excellent reasoning to select Decision Tree Classifier as best chosen model.

Student is able to clearly and concisely describe how the optimal model works in layman's terms to someone who is not familiar with machine learning nor has a technical background.

Good explanation in layman's term and adding references here which can further explain Decision Tree Classifier algorithm in detail. Your explanation is simplistic and easy to understand for anyone with little to no background in Machine Learning.This post explains the Random Forest in layman' term.

The final model chosen is correctly tuned using grid search with at least one parameter using at least three settings. If the model does not need any parameter tuning it is explicitly stated with reasonable justification.

Great work applying Decision Tree Classifier with hyperparameter settings.

```
# Create the parameters list you wish to tune, using a dictionary if needed.
parameters = {'max_depth': [2, 4, 6, 8, 10], 'min_samples_split': [2, 4, 6, 8, 10], 'min_samples_leaf': [2, 4, 6, 8, 10],}
```

Student reports the accuracy and F1 score of the optimized, unoptimized, models correctly in the table provided. Student compares the final model results to previous results obtained.

Good comparison between optimized and unoptimized models scores and assessing improvements against naive predictor.

Metric	Unoptimized Model	Optimized Model
Accuracy Score	0.8184	0.8525
F-score	0.6275	0.7230

Feature Importance

Student ranks five features which they believe to be the most relevant for predicting an individual's' income. Discussion is provided for why these features were chosen.

Good selection of five features!! Very intuitive!!

- occupation
- hours_per_day
- education
- age
- capital_gain

Student correctly implements a supervised learning model that makes use of the `feature_importances_` attribute. Additionally, student discusses the differences or similarities between the features they considered relevant and the reported relevant features.

Your intuition and reasoning matched 2 off 5 features. Good observation and discussion on differences.

Student analyzes the final model's performance when only the top 5 features are used and compares this performance to the optimized model from Question 5.

Good reasoning!! If training time is constraint then we can go with the reduced features as score are slightly less but still significant.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START