UDACITY

‹ Back to Machine Learning Engineer Nanodegree

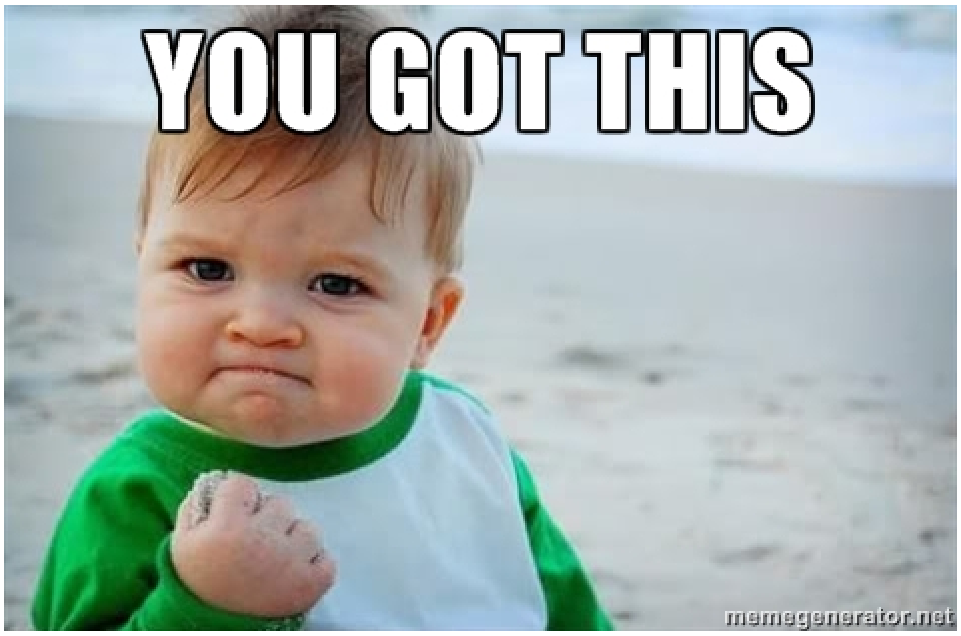
Finding Donors for CharityML

REVIEW

CODE REVIEW

HISTORY

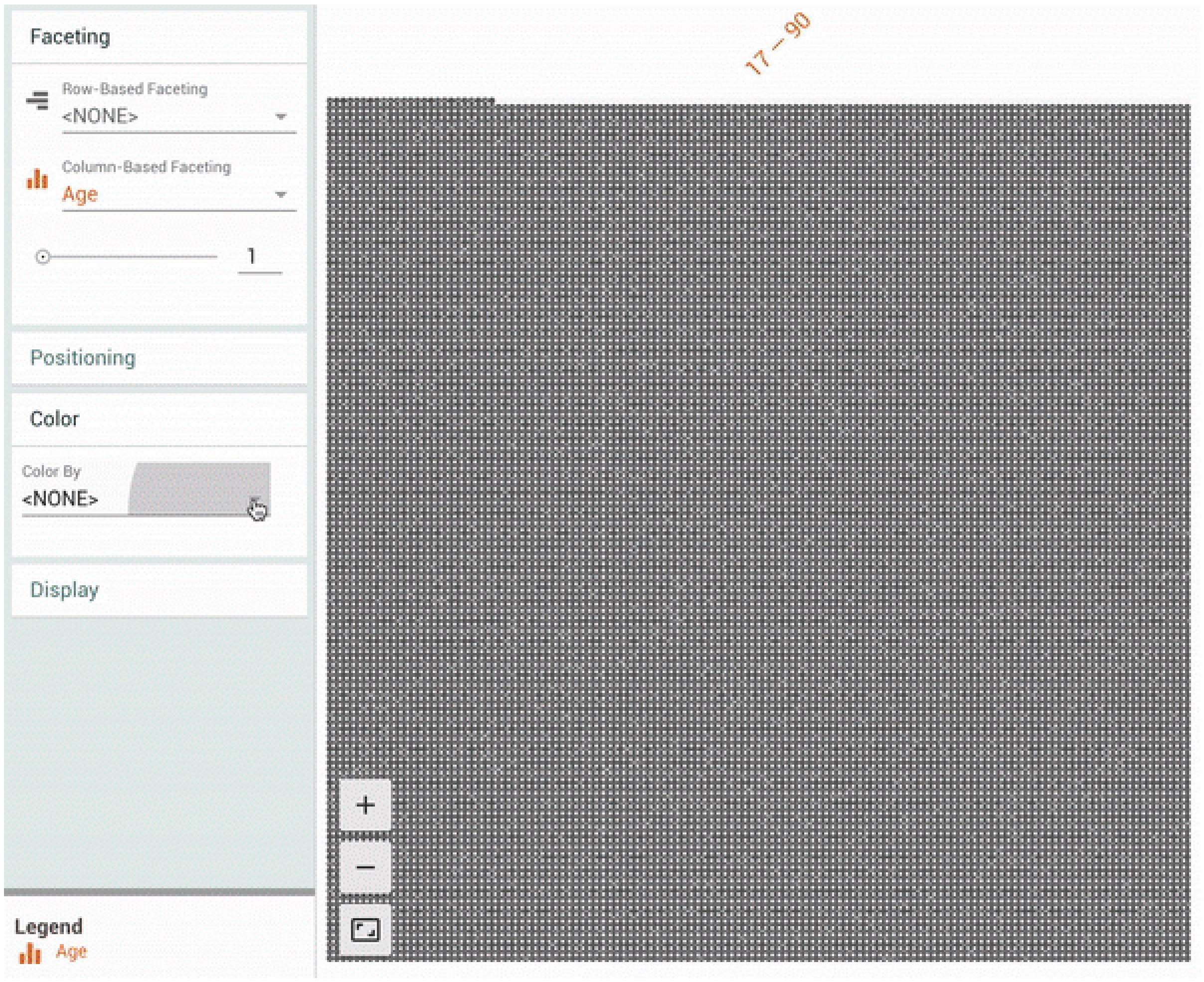
Meets Specifications



Congratulations on passing your exam!

Bonus

A partnership with Google has released an open source software to visualize machine learning datasets. And the best part is that they are using the same dataset of this project as live demo! If you want to perform some data visualization with the data of this project, go to the link https://pair-code.github.io/facets/ and play around. Here's an example from Google's blog:



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**

Bonus

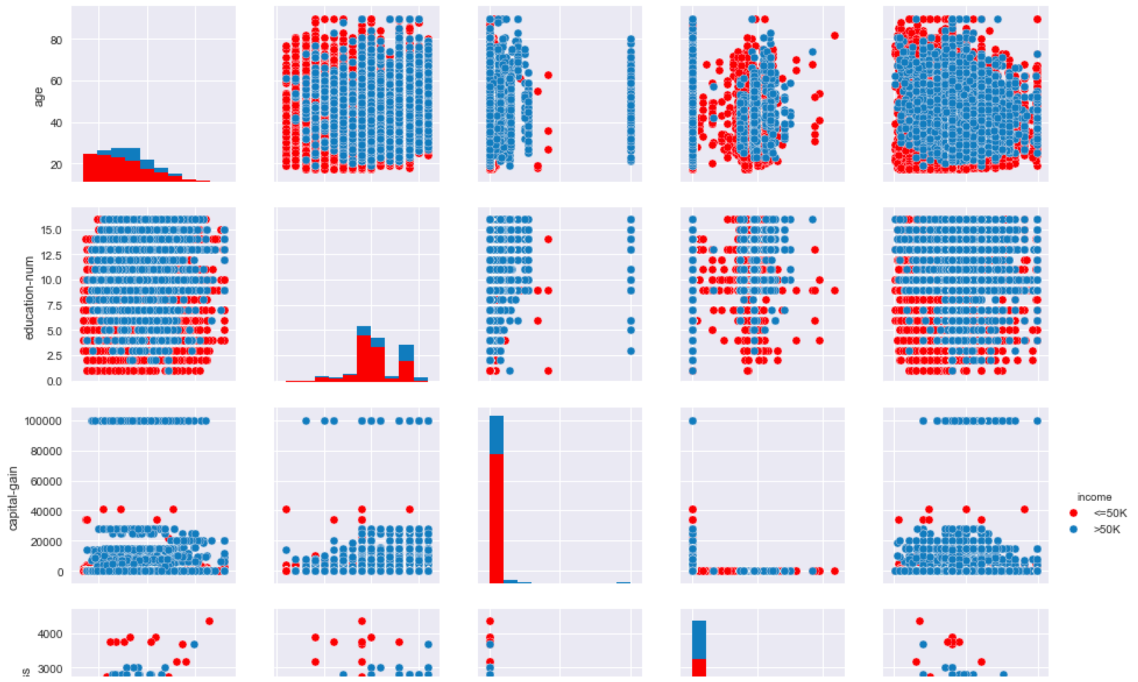
We can also get some initial statistics with the pandas method describe:

```
data.describe()
```

In addition, a very cool tool for data visualization and correlations is the pairplot of seaborn (install using `pip install seaborn`):

```
In [9]: import seaborn as sns
        sns.pairplot(data, palette='Set1', hue='income')

Out[9]: <seaborn.axisgrid.PairGrid at 0x1169b6050>
```



Preparing the Data

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

Awesome

Well done using the lambda function!

Comment

This reference provides 7 different encoding strategies. Binary encoding is a great choice for cases where the number of categories for a given feature is very high.

Evaluating Model Performance

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

Great job calculating the accuracy and the F-score for a Naive predictor!

Comment

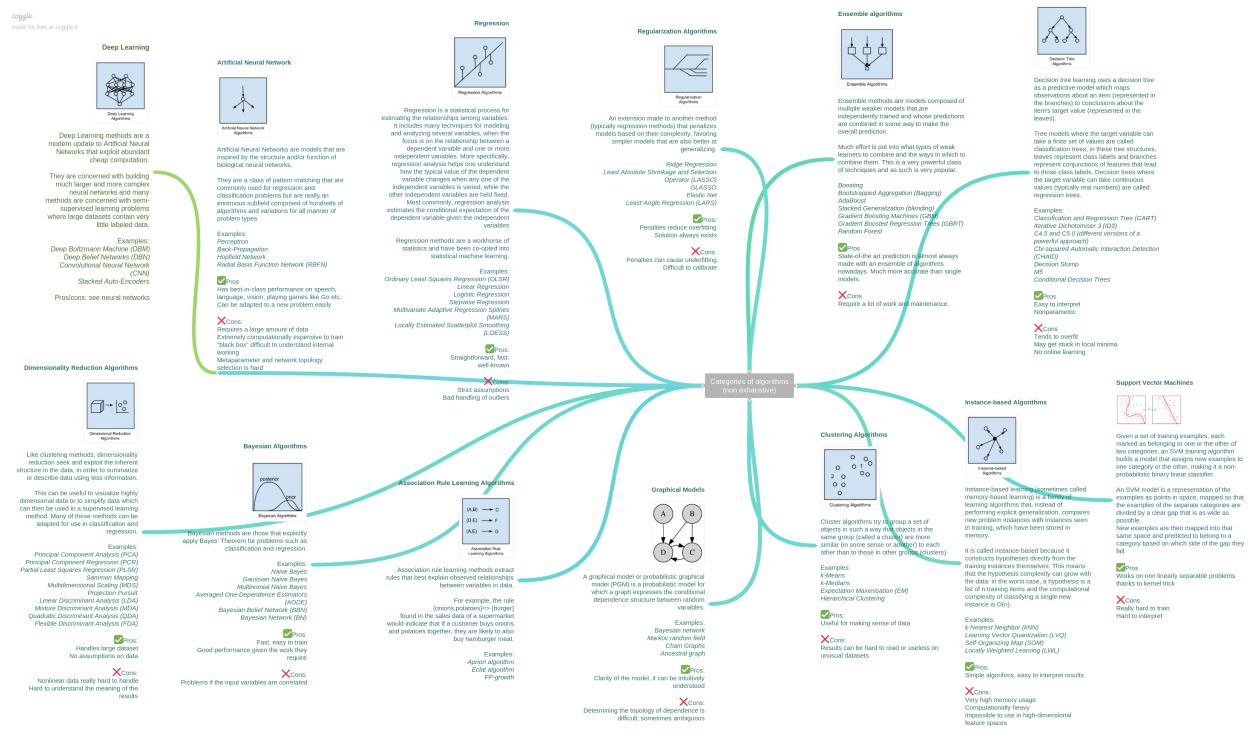
Note that the F-score is higher than the accuracy which seems counter-intuitive since the F-score is a more elaborate calculation. That happens because a value of beta = 0.5 attenuates the influence of false negatives. In other words, this value of beta weights more the positive predictions (>50K) than the negative one (<=50K).

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.

Bonus

You can also refer to this reference when choosing your estimator:


coggle
made for free at coggle.it
Categories of algorithms (non exhaustive)
Deep Learning Algorithms
Artificial Neural Network Algorithms
Regression Algorithms
Regularization Algorithms
Ensemble Algorithms
Decision Tree Algorithms
Dimensional Reduction Algorithms
posterior
prior
Bayesian Algorithms
(A,B) → C
(D,E) → F
(A,E) → G
Association Rule Learning Algorithms
A
B
D
C
Clustering Algorithms
Instance-based Algorithms

Deep Learning

Deep Learning methods are a modern update to Artificial Neural Networks that exploit abundant cheap computation.

They are concerned with building much larger and more complex neural networks and many methods are concerned with semi-supervised learning problems where large datasets contain very little labeled data.

Examples:
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders

Pros/cons: see neural networks

Artificial Neural Network

Artificial Neural Networks are models that are inspired by the structure and/or function of biological neural networks.

They are a class of pattern matching that are commonly used for regression and classification problems but are really an enormous subfield comprised of hundreds of algorithms and variations for all manner of problem types.

Examples:
Perceptron
Back-Propagation
Hopfield Network
Radial Basis Function Network (RBFN)

✓Pros
Has best-in-class performance on speech, language, vision, playing games like Go etc.
Can be adapted to a new problem easily

✗Cons:
Requires a large amount of data
Extremely computationally expensive to train
"black box" difficult to understand internal working
Metaparameter and network topology selection is hard

Regression

Regression is a statistical process for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. More specifically, regression analysis helps one understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables

Regression methods are a workhorse of statistics and have been co-opted into statistical machine learning.

Examples:
Ordinary Least Squares Regression (OLSR)
Linear Regression
Logistic Regression
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)

✓Pros:
Straightforward, fast, well-known

✗Cons:
Strict assumptions
Bad handling of outliers

Regularization Algorithms

An extension made to another method (typically regression methods) that penalizes models based on their complexity, favoring simpler models that are also better at generalizing.

Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
GLASSO
Elastic Net
Least-Angle Regression (LARS)

✓Pros:
Penalties reduce overfitting
Solution always exists

✗Cons:
Penalties can cause underfitting
Difficult to calibrate

Ensemble algorithms

Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction.

Much effort is put into what types of weak learners to combine and the ways in which to combine them. This is a very powerful class of techniques and as such is very popular.

Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (blending)
Gradient Boosting Machines (GBM)
Gradient Boosted Regression Trees (GBRT)
Random Forest

✓Pros
State-of-the art prediction is almost always made with an ensemble of algorithms nowadays. Much more accurate than single models.

✗Cons
Require a lot of work and maintenance.

Decision tree learning uses a decision tree as a predictive model which maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).

Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

Examples:
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5 and C5.0 (different versions of a powerful approach)
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
M5
Conditional Decision Trees

✓Pros
Easy to interpret
Nonparametric

✗Cons
Tends to overfit
May get stuck in local minima
No online learning

Dimensionality Reduction Algorithms

Like clustering methods, dimensionality reduction seek and exploit the inherent structure in the data, in order to summarize or describe data using less information.

This can be useful to visualize highly dimensional data or to simplify data which can then be used in a supervised learning method. Many of these methods can be adapted for use in classification and regression.

Examples:
Principal Component Analysis (PCA)
Principal Component Regression (PCR)
Partial Least Squares Regression (PLSR)
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Linear Discriminant Analysis (LDA)
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Flexible Discriminant Analysis (FDA)

✓Pros:
Handles large dataset
No assumptions on data

✗Cons:
Nonlinear data really hard to handle
Hard to understand the meaning of the results

Bayesian Algorithms

Bayesian methods are those that explicitly apply Bayes' Theorem for problems such as classification and regression.

Examples:
Naive Bayes
Gaussian Naive Bayes
Multinomial Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Bayesian Network (BN)

✓Pros:
Fast, easy to train
Good performance given the work they require

✗Cons:
Problems if the input variables are correlated

Association Rule Learning Algorithms

Association rule learning methods extract rules that best explain observed relationships between variables in data.

For example, the rule {onions,potatoes}=> {burger} found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat.

Examples:
Apriori algorithm
Eclat algorithm
FP-growth

Graphical Models

A graphical model or probabilistic graphical model (PGM) is a probabilistic model for which a graph expresses the conditional dependence structure between random variables.

Examples:
Bayesian network
Markov random field
Chain Graphs
Ancestral graph

✓Pros:
Clarity of the model, it can be intuitively understood

✗Cons:
Determining the topology of dependence is difficult, sometimes ambiguous

Clustering Algorithms

Cluster algorithms try to group a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

Examples:
k-Means
k-Medians
Expectation Maximisation (EM)
Hierarchical Clustering

✓Pros:
Useful for making sense of data

✗Cons:
Results can be hard to read or useless on unusual datasets

Instance-based Algorithms

Instance-based learning (sometimes called memory-based learning) is a family of learning algorithms that, instead of performing explicit generalization, compares new problem instances with instances seen in training, which have been stored in memory.

It is called instance-based because it constructs hypotheses directly from the training instances themselves. This means that the hypothesis complexity can grow with the data: in the worst case, a hypothesis is a list of n training items and the computational complexity of classifying a single new instance is O(n).

Examples:
k-Nearest Neighbor (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)

✓Pros:
Simple algorihms, easy to interpret results

✗Cons:
Very high memory usage
Computationally heavy
Impossible to use in high-dimensional feature spaces

Support Vector Machines

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.
New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

✓Pros
Works on non linearly separable problems thanks to kernel trick

✗Cons
Really hard to train
Hard to interpret

- This reference provides many visualization resources of machine learning algorithms. I definitely recommend it to get a visual intuition of ML techniques. My favorite is the article Gradient Boosting explained.
- This reference from sklearn compares the decision boundaries of the major classifiers.
- This map from sklearn might be also helpful in order to give guidance when choosing an estimator.
- XGBoost and Random Forest are two very popular algorithms in data science contests. This guide might be useful for tuning XGBoost.
- Three suggestion of books in case you are interested in diving deeper into the characteristics of each estimator:
    - The Elements of Statistical Learning
    - Bishop: Pattern Recognition and Machine Learning
    - Machine Learning - Tom Mitchell

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

Awesome

Everything looks great here!

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

Improving Results

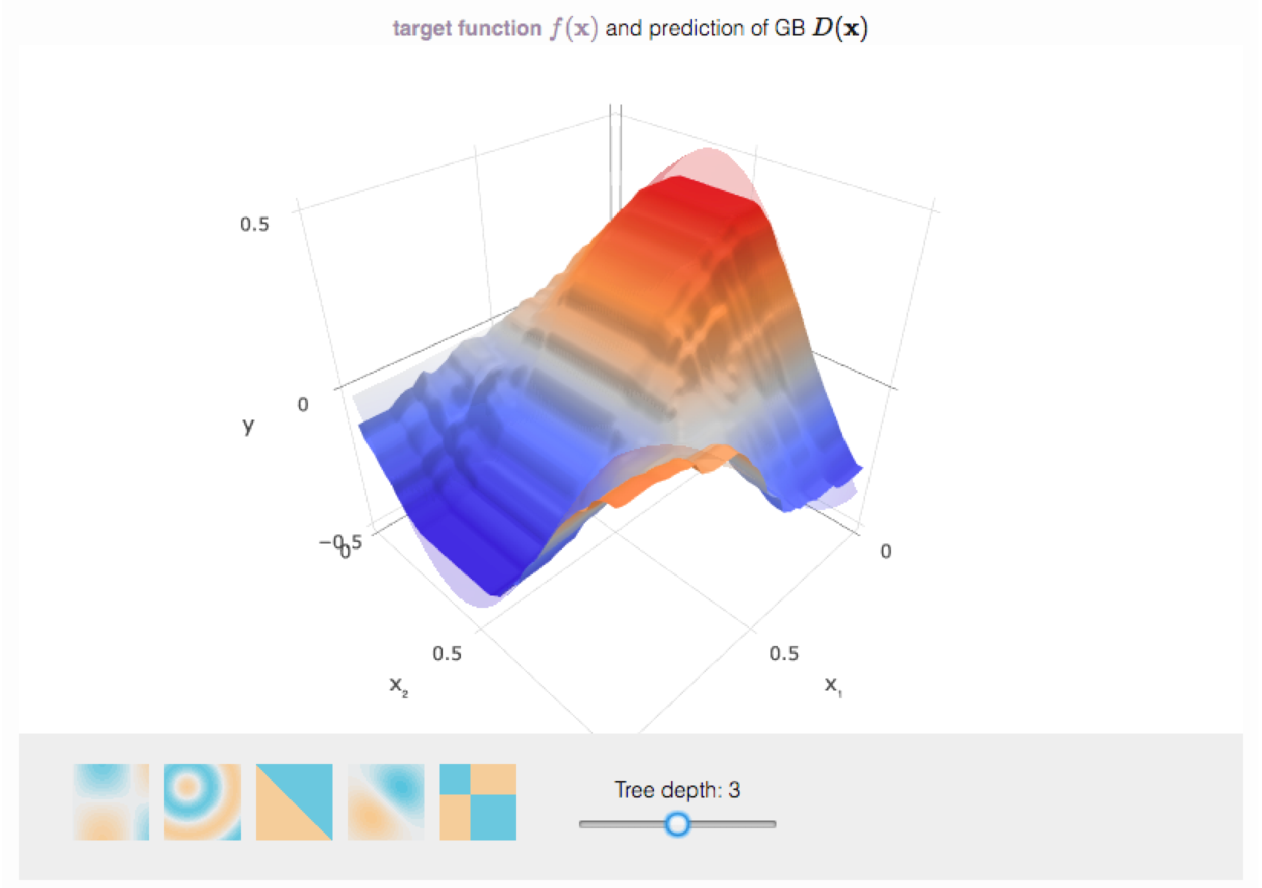
Justification is provided for which model appears to be the best to use given computational cost, model performance, and the characteristics of the data.

I agree with your choice of Gradient Boosting here! It is one of the best estimators for this project and in the analysis is the one which is leading to the highest test score. It is worth noting that it might not be fair to compare different models with their default parameters. For example, Decision Trees tend to not perform well with their default parameters while their tuned model can surpass the tuned SVM.

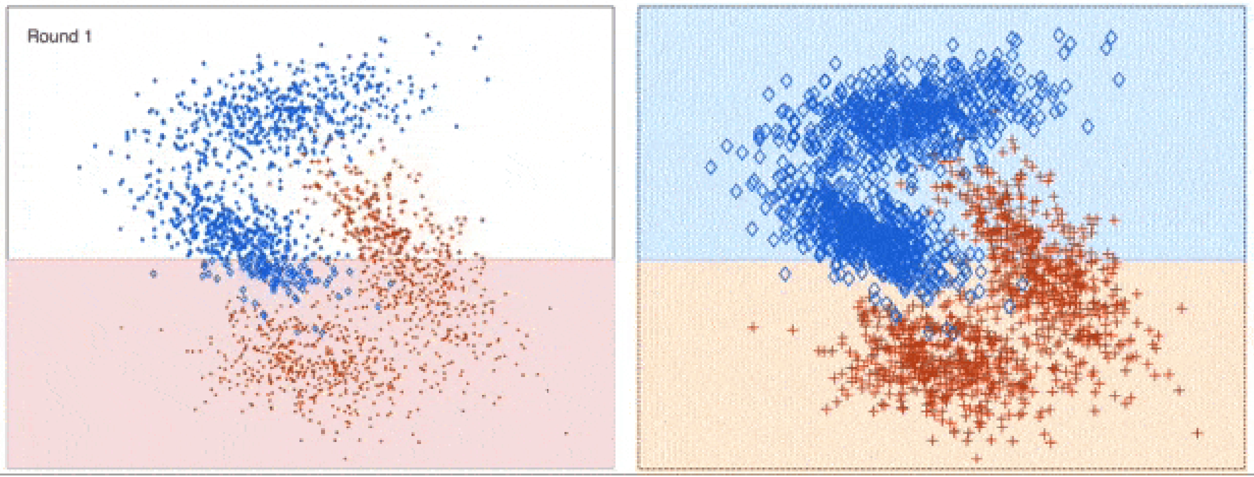
Student is able to clearly and concisely describe how the optimal model works in layman's terms to someone who is not familiar with machine learning nor has a technical background.

Bonus

This article called Gradient Boosting explained is great for visualizing GB in action:



In addition, this video shows AdaBoost (GB's cousin) in action. It might be useful to get an intuition of how this estimator works. I suggest watching it in slow motion:



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.

Bonus

Although Gradient Boosting usually give better results, AdaBoost has as advantage that we can tune its base estimator using the prefix `base_estimator__`. Here's an example:

```
# TODO: Initialize the classifier
clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(random_state=42))

# TODO: Create the parameters list you wish to tune, using a dictionary if needed.
parameters = {'n_estimators':[50,  75, 120],
              'learning_rate':[0.1, 0.5, 1.],
              'base_estimator__min_samples_split' : np.arange(2, 8, 2),
              'base_estimator__max_depth' : np.arange(1, 4, 1)
             }
```

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.

Awesome improvement in the model using grid search! Adaboost and Gradient Boosting are usually the estimators with highest scores for this project.

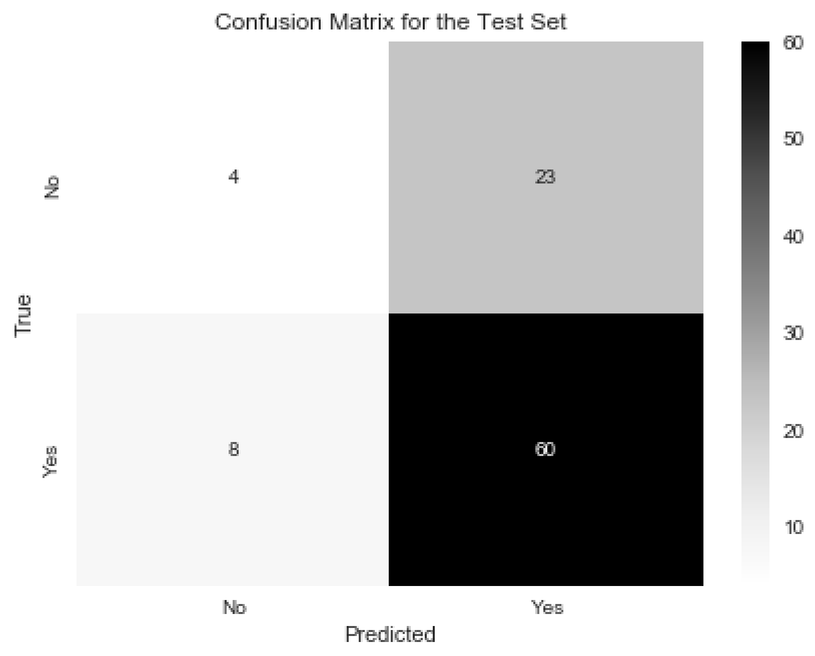
Bonus

You can also check your results with a Confusion Matrix:

```
import seaborn as sns # Install using 'pip install seaborn'
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
%matplotlib inline

cm_test = confusion_matrix(y_test, best_clf.predict(X_test))

plt.figure(figsize=(7,5))
sns.heatmap(cm_test, annot=True, cmap='Greys', xticklabels=['No', 'Yes'], yticklabel
s=['No', 'Yes'])
plt.title('Confusion Matrix for the Test Set')
plt.ylabel('True')
plt.xlabel('Predicted')
```



Bonus 2

In case you are interested in even higher scores, give a try on XGBoost (install using `pip install xgboost`). It works just like any Sklearn estimator. Here's an example to get scores above 0.75:

```
from xgboost import XGBClassifier # pip install xgboost
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer

# Initialize the classifier
clf = XGBClassifier(random_state=0)

# Create the parameters list you wish to tune, using a dictionary if needed.
# HINT: parameters = {'parameter_1': [value1, value2], 'parameter_2': [value1, value
2]}
parameters = { 'max_depth': [2, 3, 5],
               'learning_rate': [0.01, 0.1, 0.2],
               'n_estimators': [50, 100, 150]  }

# Make an fbeta_score scoring object using make_scorer()
scorer = make_scorer(fbeta_score, beta=0.5)

# Perform grid search on the classifier using 'scorer' as the scoring method using G
ridSearchCV()
grid_obj = GridSearchCV(clf, parameters, scoring=scorer, verbose=10)

# Fit the grid search object to the training data and find the optimal parameters us
ing fit()
```

```
grid_fit = grid_obj.fit(X_train, y_train)

# Get the estimator
best_clf = grid_fit.best_estimator_

print("Best parameters: ")

# Make predictions using the unoptimized and model
predictions = (clf.fit(X_train, y_train)).predict(X_test)
best_predictions = best_clf.predict(X_test)

# Report the before-and-afterscores
print("Unoptimized model\n------")
print("Accuracy score on testing data: {:.4f}".format(accuracy_score(y_test, predict
ions)))
print("F-score on testing data: {:.4f}".format(fbeta_score(y_test, predictions, beta
 = 0.5)))
print("\nOptimized Model\n------")
print("Final accuracy score on the testing data: {:.4f}".format(accuracy_score(y_tes
t, best_predictions)))
print("Final F-score on the testing data: {:.4f}".format(fbeta_score(y_test, best_pr
edictions, beta = 0.5)))
```

With more tuning, we can get scores higher than 0.78.

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.

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.

Comment

It is worth noting that each model with `feature_importances_` might return different top predictive features depending on their internal algorithm implementation.

Suggestion

You can also use the attribute `feature_importances_` from `best_clf` since it is already tuned so you will have a better choice of top 5 features:

```
importances = best_clf.feature_importances_
```

Bonus

Besides the feature importance, an alternative is to use calculate the significance of each feature using the weights of a Logistic Regression (LR) model. The significance is calculated dividing the coefficients of the LR by their standard errors (also know as Z-score). You can check more details in the Chapter 4.4.2 of The Elements of Statistical Learning. According to the reference, a Z-score higher than 2 is significant at the 5% level.

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.

Comment

An alternative strategy for reducing the number of features is to use dimensionality reduction techniques (PCA for example). Then, we could pick only the top descriptive features for training the model. You will see more details about PCA in the next module.

DOWNLOAD PROJECT

RETURN TO PATH