Titanic Survivors Data Analysis Part-2

Step 6: Model Deployment

Here we will simply deploy the various models with default parameters and see which one yields the best result. The models can further be tuned for better performance but are not in the scope of this one course. The models we will run are:

- Logistic regression
- K Nearest Neighbour
- Support Vector classifier

First, we import the necessary models

```
from sklearn.model_selection import cross_val_score

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC
```

1) Logistic Regression

```
lr = LogisticRegression(max_iter = 2000)

cv = cross_val_score(lr,X_train_scaled,y_train,cv=5)

print(cv)

print(cv.mean())
```

***Result*:**

```
[0.79775281 0.79213483 0.83146067 0.79775281 0.85310734]
0.8144416936456548
```

2) K Nearest Neighbour

```
knn = KNeighborsClassifier()

cv = cross_val_score(knn,X_train_scaled,y_train,cv=5)

print(cv)

print(cv.mean())
```

Result:

```
[0.79775281 0.79213483 0.83146067 0.79775281 0.85310734]
0.8144416936456548
```

3) Support Vector Classifier

```
svc = SVC(probability = True)

cv = cross_val_score(svc,X_train_scaled,y_train,cv=5)

print(cv)

print(cv.mean())
```

Result:

```
[0.85393258 0.82022472 0.8258427  0.80337079 0.86440678]
0.8335555132355742
```

Therefore the accuracy of the models are:

- Logistic regression: 82.2%
- K Nearest Neighbour: 81.4%
- **SVC: 83.3%**

As you can see we get decent accuracy with all our models, but the best one is SVC. And you've completed first data science project!

Another method is you can use **AutoML Tools** to Create a Model.

We really need a deeper analysis to extract more information from the data. We also need to play with the algorithms and the hyperparameters to properly tune the desired method of classification. But that's going to be a lot of work, so instead let's give the automated tooling a chance to see how much it can improve our baseline model.

We have to assembled an analysis for the Titanic dataset that includes full preprocessing, algorithm selection, hyperparameter tuning, training, predicting, and even packaging the results for submission:

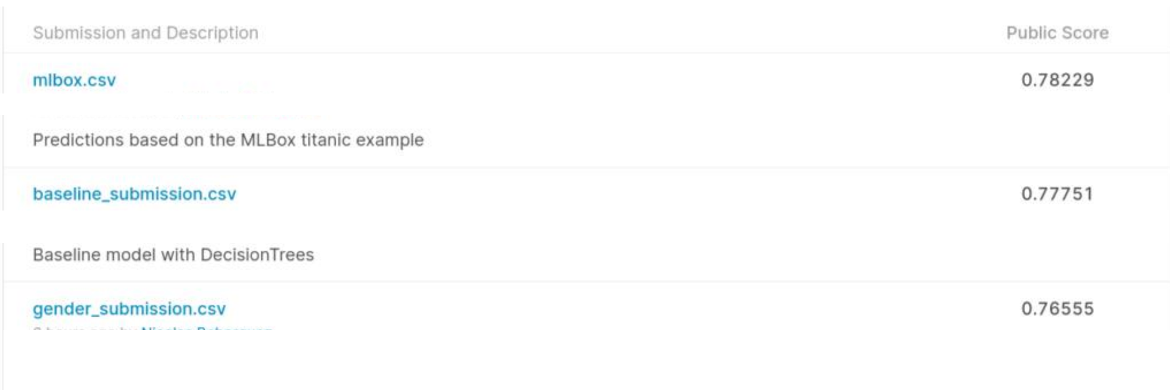
```
#1. load and preprocess data

paths = ["./data/train.csv","./data/test.csv"]

target_name = "Survived"

rd = Reader(sep = ",")
```

```
df_mlbox = rd.train_test_split(paths, target_name)


#2. removes non-stable features
dft = Drift_thresholder()
df = dft.fit_transform(df_mlbox)



#3. optimize defining a search space for each aspect of the algorithms
opt = Optimiser(scoring = "accuracy", n_folds = 5)
space = {
       'est__n_estimators':{"search":"choice","space":[150]},
       'est__colsample_bytree':{"search":"uniform","space":[0.8,0.95]},
       'est__subsample':{"search":"uniform","space":[0.8,0.95]},
       'est__max_depth':{"search":"choice","space":[5,6,7,8,9]},
       'est__learning_rate':{"search":"choice","space":[0.07]}
       }
params = opt.optimise(space, df,15)


#4. predict using the previously optimized params
prd = Predictor()
prd.fit_predict(params, df)


#5. packaging to submit
```

```
submit = pd.read_csv("./results/baseline_submission.csv",sep=',')

preds = pd.read_csv("./save/"+target_name+"_predictions.csv")

submit[target_name] =  preds[target_name+"_predicted"].values

submit.to_csv("./results/mlbox.csv", index=False)
```

In the above code:

1. Step 1 simply uses a reader to load the training and test datasets.
2. Step 2 is the most complicated, because it deals with the selection process that not only drops useless variables, but also takes care of the drifting variables. (A drifting variable changes its statistical properties from the training dataset to the test dataset).
3. Step 3 optimizes the hyperparameters by setting a search space and fitting the selected algorithm with the training data.
4. Step 4 performs the predictions and saves them in an mlbox.csv file.
5. Step 5 prepares the predictions for the result.



Submission and Description	Public Score
mlbox.csv	0.78229
Predictions based on the MLBox titanic example	
baseline_submission.csv	0.77751
Baseline model with DecisionTrees	
gender_submission.csv	0.76555

As you can see, the predictions made by the AutoML model were slightly better than the baseline model. The lesson is clear: the automatic model was better parametrized, but it still lacks the feature engineering that a human could contribute.

Though there is so much more one can do to get better results, this is more than enough to help you get started and see how you think like a data scientist. Hope this walkthrough helped you to doing the project and hope you enjoy it too.