PROJECT: PREDICTING MOVIE RENTAL DURATIONS

datalab

A DVD rental company needs your help! They want to figure out how many days a customer will rent a DVD for based on some features and has approached you for help. They want you to try out some regression models which will help predict the number of days a customer will rent a DVD for. The company wants a model which yeilds a MSE of 3 or less on a test set. The model you make will help the company become more efficient inventory planning.

The data they provided is in the csv file `rental_info.csv`. It has the following features:

- `"rental_date"`: The date (and time) the customer rents the DVD.
- `"return_date"`: The date (and time) the customer returns the DVD.
- `"amount"`: The amount paid by the customer for renting the DVD.
- `"amount_2"`: The square of `"amount"`.
- `"rental_rate"`: The rate at which the DVD is rented for.
- `"rental_rate_2"`: The square of `"rental_rate"`.
- `"release_year"`: The year the movie being rented was released.
- `"length"`: Lenght of the movie being rented, in minuites.
- `"length_2"`: The square of `"length"`.
- `"replacement_cost"`: The amount it will cost the company to replace the DVD.
- `"special_features"`: Any special features, for example trailers/deleted scenes that the DVD also has.
- `"NC-17"`, `"PG"`, `"PG-13"`, `"R"`: These columns are dummy variables of the rating of the movie. It takes the value 1 if the move is rated as the column name and 0 otherwise. For your convinience, the reference dummy has already been dropped.

```python
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
import datetime as dt

# Import any additional modules and start coding below
rental_info = pd.read_csv('rental_info.csv')
rental_info
# Convert to datetime
rental_info['rental_date'] = pd.to_datetime(rental_info['rental_date'])
rental_info['return_date'] = pd.to_datetime(rental_info['return_date'])

# create daylength column
rental_info["rental_length_days"] = (rental_info['return_date'] - rental_info['rental_date']).dt.days


# create dummy variables
rental_info["deleted_scenes"] =  np.where(rental_info["special_features"].str.contains("Deleted Scenes"), 1,0)
rental_info["behind_the_scenes"] =  np.where(rental_info["special_features"].str.contains("Behind the Scenes"), 1,
0)

# create input and target variables
X = rental_info.drop(['rental_date', 'return_date', 'rental_length_days', 'special_features'], axis=1)
y = rental_info['rental_length_days']

# split the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=9)

# instantiate a RandomForestRegressor
```

```
rf = RandomForestRegressor(random_state=9)

# Define the dictionary 'params_rf'
params_rf = {
    'n_estimators': [200, 350, 500, 700],
    'max_features': ['log2', 'sqrt'],
    'min_samples_leaf': [2,10,30]
}

# Instantiate grid_rf
grid_rf = GridSearchCV(estimator=rf,
                       param_grid=params_rf,
                       scoring='neg_mean_squared_error',
                       cv=3,
                       verbose=1,
                       n_jobs=-1)

# # fit the training set
grid_rf.fit(X_train, y_train)

# # Extract the best estimator
best_model = grid_rf.best_estimator_

# # Predict test set labels
y_pred = best_model.predict(X_test)

# # Compute rmse_test
best_mse = MSE(y_test, y_pred)

# # print the MSE
print(best_model)
print(best_mse)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
RandomForestRegressor(max_features='log2', min_samples_leaf=2, n_estimators=700,
                      random_state=9)
2.0252670320427253
```