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.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Import any additional modules and start coding below
```

```
df = pd.read_csv('rental_info.csv')
df.head()
```

	rental_date	return_date	amount	release_year	rental_rate	length	replacement_cost
0	2005-05-25 02:54:33+00:00	2005-05-28 23:40:33+00:00	2.99	2005	2.99	126	
1	2005-06-15 23:19:16+00:00	2005-06-18 19:24:16+00:00	2.99	2005	2.99	126	
2	2005-07-10 04:27:45+00:00	2005-07-17 10:11:45+00:00	2.99	2005	2.99	126	
3	2005-07-31 12:06:41+00:00	2005-08-02 14:30:41+00:00	2.99	2005	2.99	126	
4	2005-08-19 12:30:04+00:00	2005-08-23 13:35:04+00:00	2.99	2005	2.99	126	

5 rows

```
df.info()
```

Hidden output

```
df['return_date'] = pd.to_datetime(df['return_date'])
df['rental_date'] = pd.to_datetime(df['rental_date'])
df.info()
```

Hidden output

```
df['rental_length'] = df['return_date'] - df['rental_date']
df['rental_days'] = df['rental_length'].dt.days
```

```
df["deleted_scenes"] =  np.where(df["special_features"].str.contains("Deleted Scenes"), 1,0)
df["behind_the_scenes"] =  np.where(df["special_features"].str.contains("Behind the Scenes"), 1, 0)
df.head()
```

	rental_date	return_date	amount	release_year	rental_rate	length	replacement_cost
0	2005-05-25T02:54:33.000Z	2005-05-28T23:40:33.000Z	2.99	2005	2.99	126	
1	2005-06-15T23:19:16.000Z	2005-06-18T19:24:16.000Z	2.99	2005	2.99	126	
2	2005-07-10T04:27:45.000Z	2005-07-17T10:11:45.000Z	2.99	2005	2.99	126	
3	2005-07-31T12:06:41.000Z	2005-08-02T14:30:41.000Z	2.99	2005	2.99	126	
4	2005-08-19T12:30:04.000Z	2005-08-23T13:35:04.000Z	2.99	2005	2.99	126	

5 rows

```
df.info()
```

Hidden output

```
X = df.drop(['rental_days','rental_date','return_date','rental_length','special_features'], axis=1)
y = df['rental_days']
```

```
print(X.shape)
print(y.shape)
```

```
(15861, 14)
(15861,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=9)
```

```
# Perform feature selectino by choosing columns with positive coefficients

lasso = Lasso(alpha=0.3, random_state=9)
lasso.fit(X_train, y_train)
lasso_coef = lasso.coef_
X_lasso_train, X_lasso_test = X_train.iloc[:, lasso_coef > 0], X_test.iloc[:, lasso_coef > 0]
```

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_lasso_train, y_train)
lr_pred = lr.predict(X_lasso_test)
lr_mse = mean_squared_error(y_test, lr_pred)
lr_mse
```

```
4.812297241276244
```

```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor(max_depth = 4,
                           min_samples_leaf=0.1,
                           random_state = 3)
dt.fit(X_train, y_train)
dt_pred = dt.predict(X_test)
dt_mse = mean_squared_error(y_test, dt_pred)
dt_mse
```

```
3.2717707577851667
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV

param_dist = {'n_estimators': np.arange(1,101,1),
              'max_depth': np.arange(1,11,1)}

rf = RandomForestRegressor()
random_search = RandomizedSearchCV(rf,
                                    param_distributions = param_dist,
                                    cv=5,
                                    random_state=9)
random_search.fit(X_train, y_train)

hyper_params = random_search.best_params_

rf = RandomForestRegressor(n_estimators = hyper_params['n_estimators'],
                           max_depth = hyper_params['max_depth'],
                           random_state=9)

rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_pred)
rf_mse
```

```
2.225667528098759
```

```
best_model = rf
```