
Applications of
Machine Learning
Homework Six
Matt Ferguson

Requirements

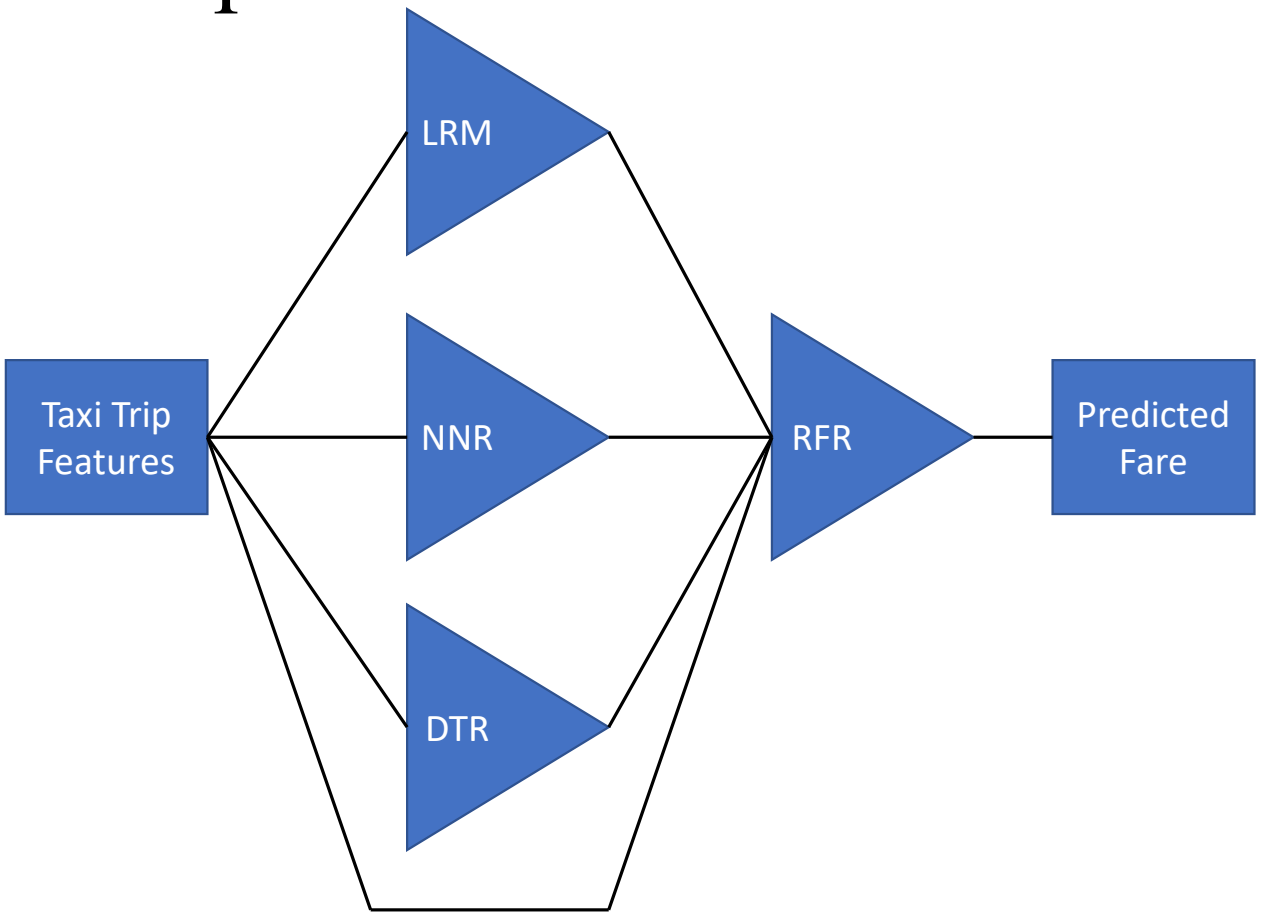
- Develop a two-stage regression model to predict taxicab fare
- Describe and graph multistage model
- Model performance metrics
- Plot model learning curve
- Plot predicted and actual fare
- Attach developed machine learning code

Multistage Model Graphic

We see on the right a graphic representing the developed multistage model.

Our taxi trip features are preprocessed and fed into a linear regression model (LRM), a neural network regressor (NNR) and a decision tree regressor (DTR) which are arranged in parallel.

The predicted fare output of the LRM, NNR, and DTR are used as features in addition to the taxi trip features for training the final stage of the model comprised of a random forest regressor (RFR).



Multistage Model Development

Taxi trip data is read into python and stored in a pandas data frame. Unusable features are dropped from the data frame along with non-target price features. One hot encoding is performed on trip pickup and drop off categories. Pickup and dropoff times are leveraged to extract date features. Normalization is applied to the data frame.

A linear regression model, a neural network model, and a decision tree regressor are trained using the preprocessed taxi trip data to predict fare amount.

Finally, a random forest regressor is trained using the outputs of the three fare predictions along with the original preprocessed feature set. Note that library imports are omitted from the code snipping presented on the right.

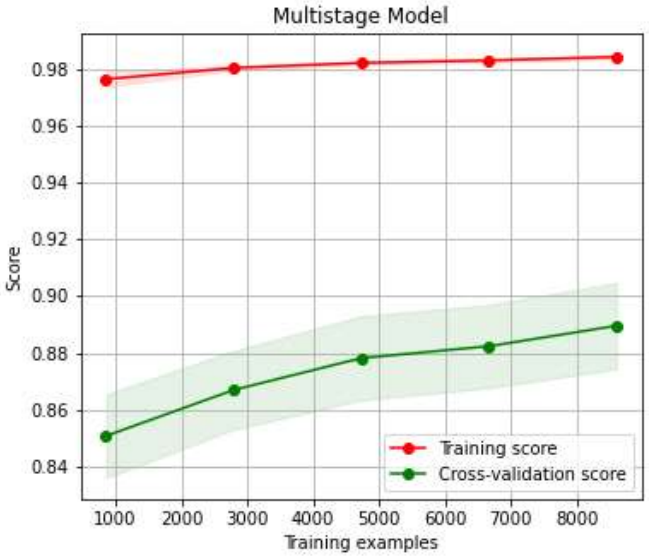
```
df=pd.read_excel('Taxi_Trip_Data.xlsx')
df=df.drop(['store_and_fwd_flag','PULocationID','DOLocationID','fare_amount','extra','mta_tax','tip_amount','tolls_amount'], axis=1)

df=pd.get_dummies(df,'DOBorough')
df=pd.get_dummies(df,'PUBorough')

df['PU_year'] = df['lpep_pickup_datetime'].dt.year
df['PU_month'] = df['lpep_pickup_datetime'].dt.month
df['PU_day'] = df['lpep_pickup_datetime'].dt.day
df['PU_hour'] = df['lpep_pickup_datetime'].dt.hour
df['PU_minute'] = df['lpep_pickup_datetime'].dt.minute
df['PU_dayofweek'] = df['lpep_pickup_datetime'].dt.dayofweek

df['DO_year'] = df['lpep_dropoff_datetime'].dt.year
df['DO_month'] = df['lpep_dropoff_datetime'].dt.month
df['DO_day'] = df['lpep_dropoff_datetime'].dt.day
df['DO_hour'] = df['lpep_dropoff_datetime'].dt.hour
df['DO_minute'] = df['lpep_dropoff_datetime'].dt.minute
df['DO_dayofweek'] = df['lpep_dropoff_datetime'].dt.dayofweek

df=df.drop(['lpep_pickup_datetime','lpep_dropoff_datetime'], axis=1)

corr=df.corr()
df=(df-df.min())/(df.max()-df.min())
x=np.array(df.drop(['total_amount'], axis=1))
y=np.array(df['total_amount'])
(x_train, x_test, y_train, y_test) = sk.model_selection.train_test_split(x, y, test_size=0.3, random_state=22222)

# Linear Regression
linear_regression_model = linear_model.LinearRegression()
linear_regression_model.fit(x_train,y_train)
y_pred_lrm=linear_regression_model.predict(x_test)
R2_lrm=linear_regression_model.score(x_test,y_test)
mae_lrm=metrics.mean_absolute_error(y_test,y_pred_lrm)
evs_lrm=metrics.explained_variance_score(y_test,y_pred_lrm)
mse_lrm=metrics.mean_squared_error(y_test,y_pred_lrm)

# Neural Network Regression
neural_network_model = neural_network.MLPRegressor(hidden_layer_sizes=(10,10,10), activation='identity',
learning_rate='adaptive', tol=0.0001, solver='adam', alpha=0.0001, early_stopping=True, validation_fraction=0.1)
neural_network_model.fit(x_train, y_train)
y_pred_cnn=neural_network_model.predict(x_test)
R2_cnn=neural_network_model.score(x_test,y_test)
mae_cnn=metrics.mean_absolute_error(y_test, y_pred_cnn)
evs_cnn=metrics.explained_variance_score(y_test,y_pred_cnn)
mse_cnn=metrics.mean_squared_error(y_test,y_pred_cnn)

# Regression Decision Tree
regression_tree_model=tree.DecisionTreeRegressor(max_depth=3)
regression_tree_model.fit(x_train,y_train)
y_pred_rtm=regression_tree_model.predict(list(x_test),list(y_test))
R2_rtm=metrics.r2_score(y_test,y_pred_rtm)
mae_rtm=metrics.mean_absolute_error(y_test,y_pred_rtm)
evs_rtm=metrics.explained_variance_score(y_test,y_pred_rtm)
mse_rtm=metrics.mean_squared_error(y_test,y_pred_rtm)

x=np.column_stack((x_test,y_pred_lrm,y_pred_cnn,y_pred_rtm))
y=y_test
(x_train, x_test, y_train, y_test) = sk.model_selection.train_test_split(x, y, test_size=0.3, random_state=22222)

#Final Stage
random_forest_model=ensemble.RandomForestRegressor(random_state=22222)
plot_learning_curve(estimator=random_forest_model,X=x_train,y=y_train,cv=5, title="Multistage Model")
random_forest_model.fit(x_train,y_train)
y_pred_rfr=random_forest_model.predict(x_test)
R2_rfr=metrics.r2_score(y_test,y_pred_rfr)
mae_rfr=metrics.mean_absolute_error(y_test,y_pred_rfr)
mse_rfr=metrics.mean_squared_error(y_test,y_pred_rfr)
evs_rfr=metrics.explained_variance_score(y_test, y_pred_rfr)
```

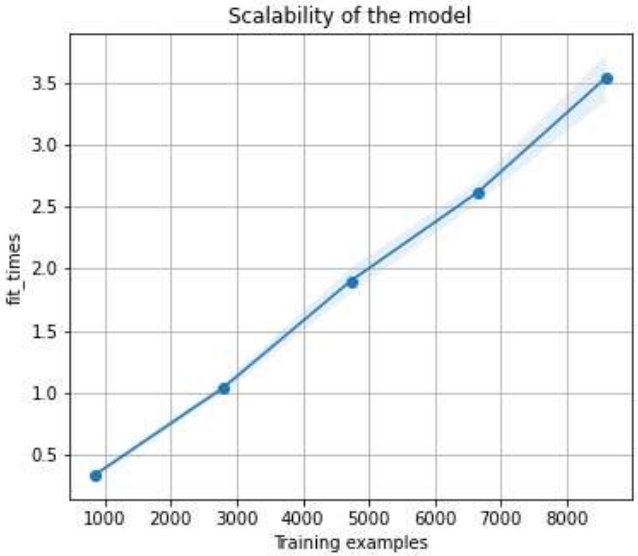
Model Performance Metrics

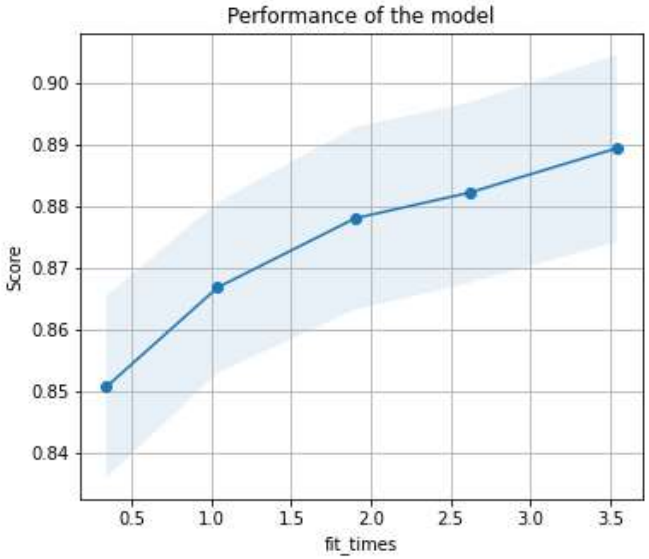
	Coefficient of Determination	Mean Square Error	Mean Absolute Error	Evaluation Score
Neural Network Regressor	0. 803	0.000130	0.00632	0.804
Linear Regression	0.807	0.000126	0.00634	0.807
Regression Decision Tree	0.793	0.000135	0.00657	0.793
Random Forest Regressor	0.873	0.000085	0.00488	0.873

Our random forest regressor had the best overall performance, with the lowest error and the highest goodness of fit. This makes sense intuitively as the random forest regressor is the final stage of our model and has access to the outputs of the other three models to leverage when predicting fares.

Learning Curve Plot







Function to plot learning curve sourced from:
https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

Predicted and Actual Fare Plot