Case Study 7, Part 2 Homework: Exercises 1-4

Exercise 1

0/1 point (graded)

In Exercise 1, we will instantiate regression and classification models. Code is provided that prepares the covariates and outcomes we will use for data analysis.

Instructions

- Instantiate `LinearRegression()`, `LogisticRegression()`, `RandomForestRegressor()`, and `RandomForestClassifier()` objects, and assign them to `linear_regression`, `logistic_regression`, `forest_regression`, and `forest_classifier`, respectively.
- For the random forests models, specify `max_depth=4` and `random_state=0`.

Fill in the code below:

```
      # Define all covariates and outcomes from `df`.
regression_target = 'revenue'
classification_target = 'profitable'
all_covariates = ['budget', 'popularity', 'runtime', 'vote_count', 'vote_average',
'Action', 'Adventure', 'Fantasy',
                  'Science Fiction', 'Crime', 'Drama', 'Thriller', 'Animation',
'Family', 'Western', 'Comedy', 'Romance',
                  'Horror', 'Mystery', 'War', 'History', 'Music', 'Documentary',
'TV Movie', 'Foreign']

regression_outcome = df[regression_target]
classification_outcome = df[classification_target]
covariates = df[all_covariates]

# Instantiate all regression models and classifiers.
linear_regression =
logistic_regression =
forest_regression =
forest_classifier =
```

What is the correct way to instantiate the `RandomForestRegressor` object?

```
forest_regression = RandomForest("regression")
```

```
forest_regression = RandomForest(max_depth=4, random_state=0,
"regression")
```

```
forest_regression = RandomForestRegressor(max_depth=4, random_state=0)
```

correct

```
forest_regression = RandomForestRegressor()
```

Code = [

```
inear_regression = LinearRegression()
logistic_regression = LogisticRegression()
forest_regression = RandomForestRegressor(max_depth=4, random_state=0)
forest_classifier = RandomForestClassifier(max_depth=4, random_state=0)
```

]

Exercise 2

1/1 point (graded)

In Exercise 2, we will create two functions that compute a model's score. For regression models, we will use correlation as the score. For classification models, we will use accuracy as the score.

Instructions

- Define a function called `correlation` with arguments `estimator`, `X`, and `y`. The function should compute the correlation between the observed outcome `y` and the outcome predicted by the model.
    - To obtain predictions, the function should first use the `fit` method of `estimator` and then use the `predict` method from the fitted object.
    - The function should return the first argument from `r2_score` comparing `predictions` and `y`.
- Define a function called `accuracy` with the same arguments and code, substituting `accuracy_score` for `r2_score`.

What is the correct way to obtain predictions using the model estimator?

```
predictions = estimator.fit(X,y).predict(X)
```

correct

```
predictions = model.fit().predict(X)
```

```
predictions = estimator.predict(X,y)
```

```
predictions = estimator.fit(X,y,"predict")
```

Code = [

```
 def correlation(estimator, X, y):
    predictions = estimator.fit(X, y).predict(X)
    return r2_score(y, predictions)

def accuracy(estimator, X, y):
    predictions = estimator.fit(X, y).predict(X)
    return accuracy_score(y, predictions)
```

]

Exercise 3

1/1 point (graded)

In Exercise 3, we will compute the cross-validated performance for the linear and random forest regression models.

Instructions

- Call `cross_val_score` using `linear_regression` and `forest_regression` as models. Store the output as `linear_regression_scores` and `forest_regression_scores`, respectively.

    - Set the parameters `cv=10` to use 10-fold cross-validation and `scoring=correlation` to use the `correlation` function defined in Exercise 2.
- Plotting code has been provided to compare the performance of the two models. Use `plt.show()` to plot the correlation between actual and predicted revenue for each cross-validation fold using the linear and random forest regression models.
- Consider which of the two models exhibits a better fit.

Here is the code framework for you to use:

```
      # Determine the cross-validated correlation for 
linear and random forest models.

# Plot Results
plt.axes().set_aspect('equal', 'box')
plt.scatter(linear_regression_scores, 
forest_regression_scores)
plt.plot((0, 1), (0, 1), 'k-')

plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Linear Regression Score")
plt.ylabel("Forest Regression Score")

# Show the plot.
```

Which model is performing better for regression?

Linear Regression

Random Forest

Correct

Code = [

```
linear_regression_scores = cross_val_score(linear_regression, covariates, 
regression_outcome, cv=10, scoring=correlation)
forest_regression_scores = cross_val_score(forest_regression, covariates, 
regression_outcome, cv=10, scoring=correlation)

plt.axes().set_aspect('equal', 'box')
plt.scatter(linear_regression_scores, forest_regression_scores)
plt.plot((0, 1), (0, 1), 'k-')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Linear Regression Score")
plt.ylabel("Forest Regression Score")

plt.show()
```

]

Exercise 4

0/1 point (graded)

In Exercise 4, we will compute the cross-validated performance for the linear and random forest classification models.

Instructions

- Call `cross_val_score` using `logistic_regression` and `forest_classifier` as models. Store the output as `logistic_regression_scores` and `forest_classification_scores`, respectively.
    - Set the parameters `cv=10` to use 10-fold cross-validation and `scoring=correlation` to use the `accuracy` function defined in Exercise 2.

- Plotting code has been provided to compare the performance of the two models. Use `plt.show()` to plot the accuracy of predicted profitability for each cross-validation fold using the logistic and random forest classification models.
- Consider which of the two models exhibits a better fit.

Here is the code framework for you to use:

```
      # Determine the cross-validated accuracy for logistic and random forest models.

# Plot Results
plt.axes().set_aspect('equal', 'box')
plt.scatter(logistic_regression_scores, forest_classification_scores)
plt.plot((0, 1), (0, 1), 'k-')

plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Linear Classification Score")
plt.ylabel("Forest Classification Score")

# Show the plot.
```

Which model is performing better for classification?

Linear Regression

Random Forest
correct

Code = [

```
logistic_regression_scores = cross_val_score(logistic_regression, covariates, 
classification_outcome, cv=10, scoring=accuracy)
forest_classification_scores = cross_val_score(forest_classifier, covariates, 
classification_outcome, cv=10, scoring=accuracy)

plt.axes().set_aspect('equal', 'box')
plt.scatter(logistic_regression_scores, forest_classification_scores)
plt.plot((0, 1), (0, 1), 'k-')

plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Linear Classification Score")
plt.ylabel("Forest Classification Score")

plt.show()
```

]

Exercise 5

0/1 point (graded)

In Exercise 5, we will rerun the regression analysis for this subsetted dataset.

Instructions

- Define `positive_revenue_df` as the subset of movies in `df` with `revenue` greater than zero.
- Code is provided below that creates new instances of model objects. Replace all instances of `df` with `positive_revenue_df`, and run the given code.

Use this code to get started:

```
        positive_revenue_df = 
```

```
# Replace the dataframe in the following code, and run.
regression_outcome = df[regression_target]
```

```
classification_outcome = df[classification_target]
covariates = df[all_covariates]

# Reinstantiate all regression models and classifiers.
linear_regression = LinearRegression()
logistic_regression = LogisticRegression()
forest_regression = RandomForestRegressor(max_depth=4,
random_state=0)
forest_classifier = RandomForestClassifier(max_depth=4,
random_state=0)
linear_regression_scores =
cross_val_score(linear_regression, covariates,
regression_outcome, cv=10, scoring=correlation)
forest_regression_scores =
cross_val_score(forest_regression, covariates,
regression_outcome, cv=10, scoring=correlation)
logistic_regression_scores =
cross_val_score(logistic_regression, covariates,
classification_outcome, cv=10, scoring=accuracy)
forest_classification_scores =
cross_val_score(forest_classifier, covariates,
classification_outcome, cv=10, scoring=accuracy)
```

What is the mean of the 10 cross validation scores for random forest regression?

Answer = [0.778]

Code =[

```
positive_revenue_df = df[df["revenue"] > 0]

regression_outcome = positive_revenue_df[regression_target]
classification_outcome = positive_revenue_df[classification_target]
covariates = positive_revenue_df[all_covariates]
```

```
linear_regression = LinearRegression()
logistic_regression = LogisticRegression()
forest_regression = RandomForestRegressor(max_depth=4, random_state=0)
forest_classifier = RandomForestClassifier(max_depth=4, random_state=0)
linear_regression_scores = cross_val_score(linear_regression, covariates, 
regression_outcome, cv=10, scoring=correlation)
forest_regression_scores = cross_val_score(forest_regression, covariates, 
regression_outcome, cv=10, scoring=correlation)
logistic_regression_scores = cross_val_score(logistic_regression, covariates, 
classification_outcome, cv=10, scoring=accuracy)
forest_classification_scores = cross_val_score(forest_classifier, covariates, 
classification_outcome, cv=10, scoring=accuracy)

np.mean(forest_regression_scores)
0.778968312886712
```

]

Exercise 6

0.63/1 point (graded)

In Exercise 6, we will compute the cross-validated performance for the logistic regression and random forest classification models for positive revenue movies only.

Instructions

- Call `cross_val_score` using `logistic_regression` and `forest_classifier` as models. Store the output as `logistic_regression_scores` and `forest_classification_scores`, respectively.
    - Set the parameters `cv=10` to use 10-fold cross-validation and `scoring=correlation` to use the `accuracy` function defined in Exercise 2.
- Plotting code has been provided to compare the performance of the two models. Use `plt.show()` to plot the correlation between actual and predicted

revenue for each cross-validation fold using the logistic regression and random forest classification models. Consider which of the two models exhibits a better fit. Is this result different from what we observed when considering all movies?

- Code is provided for you that prints the importance of each covariate in predicting revenue using the random forest classifier. Consider which variables are the most important.

Here is the code to get you started:

```
       # Determine the cross-validated accuracy for logistic and random forest models.

# Plot Results
plt.axes().set_aspect('equal', 'box')
plt.scatter(logistic_regression_scores, forest_classification_scores)
plt.plot((0, 1), (0, 1), 'k-')

plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Linear Classification Score")
plt.ylabel("Forest Classification Score")

# Show the plot.

# Print the importance of each covariate in the random forest classification.
forest_classifier.fit(positive_revenue_df[all_covariates], classification_outcome)
```

```
sorted(list(zip(all_covariates,
forest_classifier.feature_importances_)), key=lambda
tup: tup[1])
```

What are the 3 most important variables for predicting revenue in the random forest model?

Select THREE.

`Action`

`Adventure`

`budget`

`Crime`

`popularity`
correct

`runtime`

`vote_average`
correct

`vote_count`
correct

Code = [

```
logistic_regression_scores = cross_val_score(logistic_regression, covariates,
classification_outcome, cv=10, scoring=accuracy)
forest_classification_scores = cross_val_score(forest_classifier, covariates,
classification_outcome, cv=10, scoring=accuracy)

plt.axes().set_aspect('equal', 'box')
plt.scatter(logistic_regression_scores, forest_classification_scores)
plt.plot((0, 1), (0, 1), 'k-')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Linear Classification Score")
plt.ylabel("Forest Classification Score")

plt.show();

forest_classifier.fit(positive_revenue_df[all_covariates],
positive_revenue_df[classification_target])
sorted(list(zip(all_covariates, forest_classifier.feature_importances_)),
key=lambda tup: tup[1])
[('TV Movie', 0.0),
 ('Horror', 0.001715202327676785),
 ('Animation', 0.0019388197444951466),
 ('Comedy', 0.002257468989929606 5),
 ('Foreign', 0.0022801352325337114),
 ('Documentary', 0.002846458591904433),
 ('Romance', 0.0031608732977368944),
 ('Thriller', 0.0035569898966812397),
 ('Mystery', 0.004282452349394276),
 ('Music', 0.004308655018573079),
 ('Fantasy', 0.0051937079152913745),
 ('Western', 0.005480591973153852),
 ('Family', 0.0066609392542522055),
 ('Crime', 0.006772395781754328),
 ('History', 0.006793172805113654),
 ('Action', 0.0073412694021133835),
 ('Adventure', 0.007596959755592538),
 ('Science Fiction', 0.010816587516514861),
 ('War', 0.011275947022575308),
 ('Drama', 0.023093574562804687),
 ('runtime', 0.04154729351420867),
 ('budget', 0.08765680648089587),
 ('vote_average', 0.10261105225795153),
 ('popularity', 0.2811360280003983),
 ('vote_count', 0.36967661830845444)]
```

]