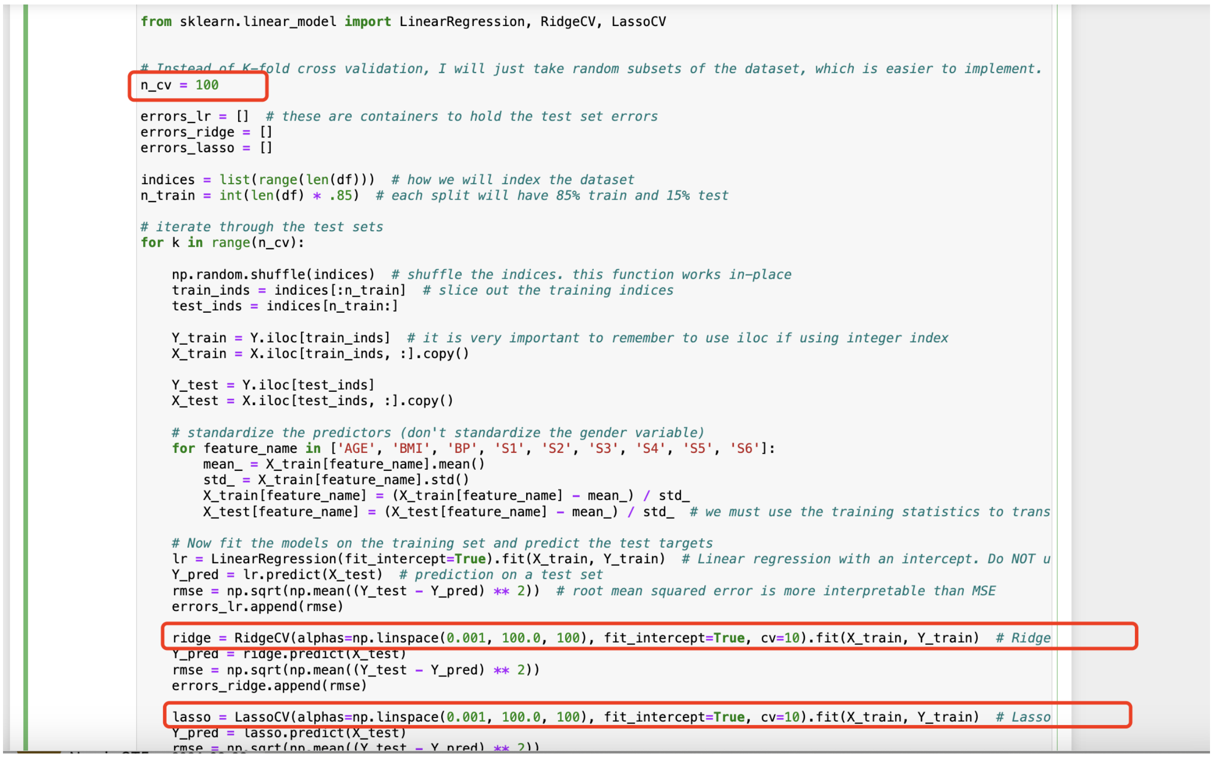
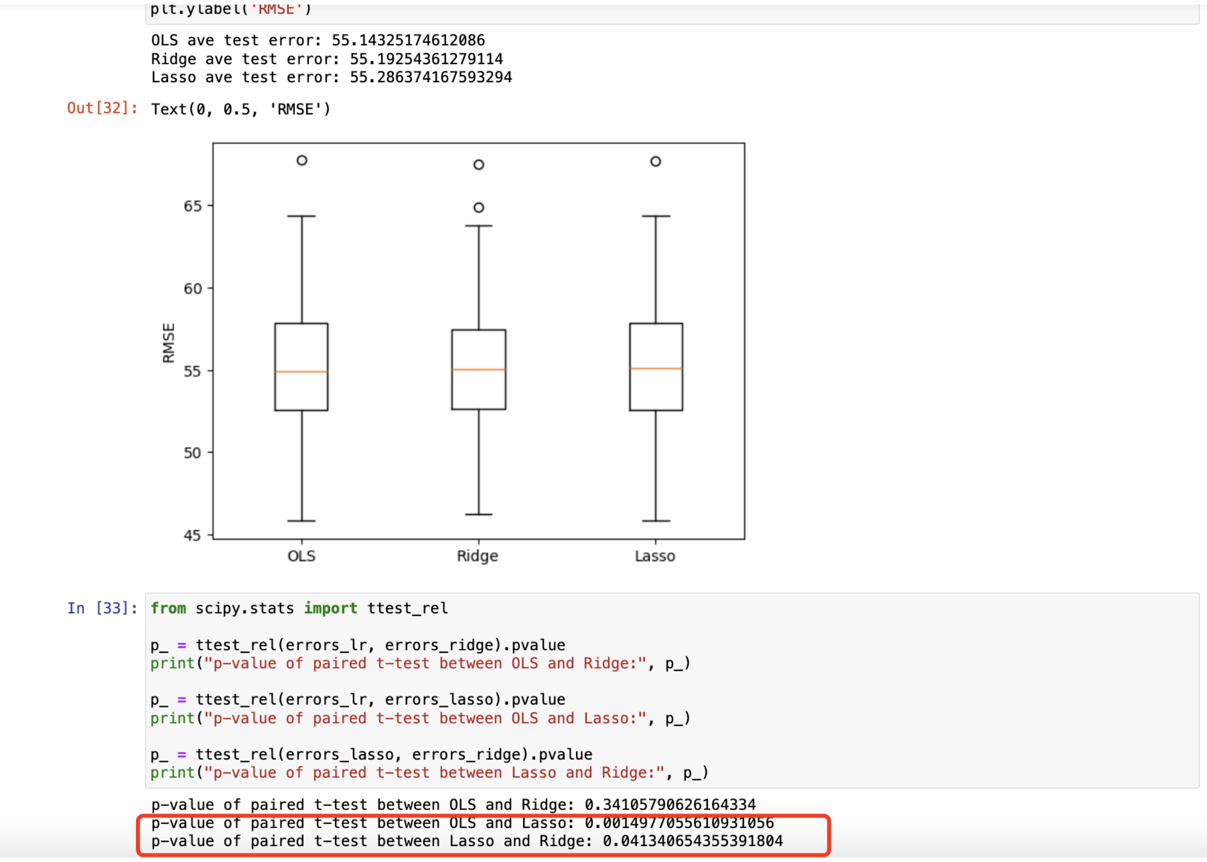
Homework 2

Problem 1



```
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV

# Instead of K-fold cross validation, I will just take random subsets of the dataset, which is easier to implement.
n_cv = 100

errors_lr = []  # these are containers to hold the test set errors
errors_ridge = []
errors_lasso = []

indices = list(range(len(df)))  # how we will index the dataset
n_train = int(len(df) * .85)  # each split will have 85% train and 15% test

# iterate through the test sets
for k in range(n_cv):

    np.random.shuffle(indices)  # shuffle the indices. this function works in-place
    train_inds = indices[:n_train]  # slice out the training indices
    test_inds = indices[n_train:]

    Y_train = Y.iloc[train_inds]  # it is very important to remember to use iloc if using integer index
    X_train = X.iloc[train_inds, :].copy()

    Y_test = Y.iloc[test_inds]
    X_test = X.iloc[test_inds, :].copy()

    # standardize the predictors (don't standardize the gender variable)
    for feature_name in ['AGE', 'BMI', 'BP', 'S1', 'S2', 'S3', 'S4', 'S5', 'S6']:
        mean_ = X_train[feature_name].mean()
        std_ = X_train[feature_name].std()
        X_train[feature_name] = (X_train[feature_name] - mean_) / std_
        X_test[feature_name] = (X_test[feature_name] - mean_) / std_  # we must use the training statistics to trans

    # Now fit the models on the training set and predict the test targets
    lr = LinearRegression(fit_intercept=True).fit(X_train, Y_train)  # Linear regression with an intercept. Do NOT u
    Y_pred = lr.predict(X_test)  # prediction on a test set
    rmse = np.sqrt(np.mean((Y_test - Y_pred) ** 2))  # root mean squared error is more interpretable than MSE
    errors_lr.append(rmse)

    ridge = RidgeCV(alphas=np.linspace(0.001, 100.0, 100), fit_intercept=True, cv=10).fit(X_train, Y_train)  # Ridge
    Y_pred = ridge.predict(X_test)
    rmse = np.sqrt(np.mean((Y_test - Y_pred) ** 2))
    errors_ridge.append(rmse)

    lasso = LassoCV(alphas=np.linspace(0.001, 100.0, 100), fit_intercept=True, cv=10).fit(X_train, Y_train)  # Lasso
    Y_pred = lasso.predict(X_test)
    rmse = np.sqrt(np.mean((Y_test - Y_pred) ** 2))
```

Changed number of splits in cross-validation from 20 to 100, and for number of folds in LassoCV and RidgeCV, since 10 is already a large enough number, I did not further increment it.



```
plt.ylabel('RMSE')
```

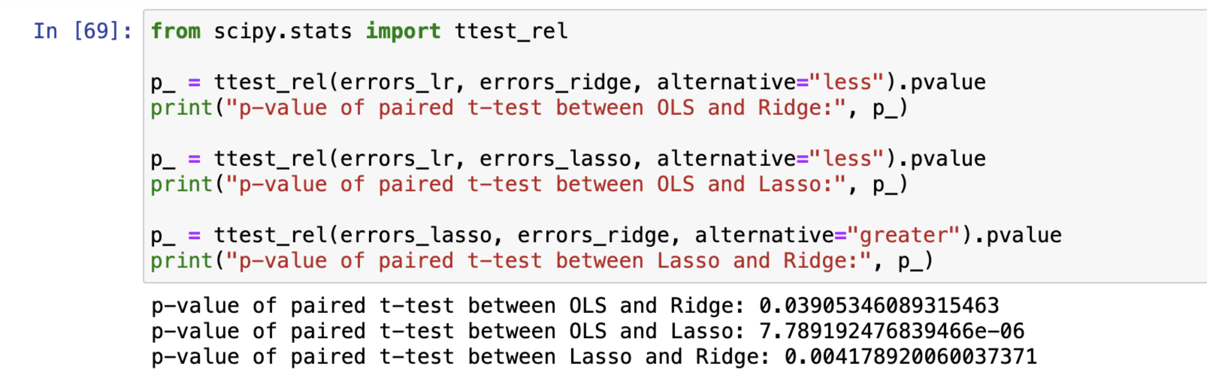
```
OLS ave test error: 55.14325174612086
Ridge ave test error: 55.19254361279114
Lasso ave test error: 55.286374167593294
```

```
Out[32]: Text(0, 0.5, 'RMSE')
```

```
In [33]: from scipy.stats import ttest_rel

         p_ = ttest_rel(errors_lr, errors_ridge).pvalue
         print("p-value of paired t-test between OLS and Ridge:", p_)

         p_ = ttest_rel(errors_lr, errors_lasso).pvalue
         print("p-value of paired t-test between OLS and Lasso:", p_)

         p_ = ttest_rel(errors_lasso, errors_ridge).pvalue
         print("p-value of paired t-test between Lasso and Ridge:", p_)
```

```
p-value of paired t-test between OLS and Ridge: 0.34105790626164334
p-value of paired t-test between OLS and Lasso: 0.0014977055610931056
p-value of paired t-test between Lasso and Ridge: 0.041340654355391804
```

Also note that by default ttest_rel uses a two-sided test and there's a parameter that allows us to set the alternative hypothesis, which is that we can test if the first mean is less/greater than the other mean on a statistically significant level. For example:



```
In [69]: from scipy.stats import ttest_rel

         p_ = ttest_rel(errors_lr, errors_ridge, alternative="less").pvalue
         print("p-value of paired t-test between OLS and Ridge:", p_)

         p_ = ttest_rel(errors_lr, errors_lasso, alternative="less").pvalue
         print("p-value of paired t-test between OLS and Lasso:", p_)

         p_ = ttest_rel(errors_lasso, errors_ridge, alternative="greater").pvalue
         print("p-value of paired t-test between Lasso and Ridge:", p_)
```

```
p-value of paired t-test between OLS and Ridge: 0.03905346089315463
p-value of paired t-test between OLS and Lasso: 7.789192476839466e-06
p-value of paired t-test between Lasso and Ridge: 0.004178920060037371
```

This would suggest that OLS was significantly outperforming Ridge and Lasso, and Lasso was

significantly outperforming Ridge in my experiment.