FINAL REPORT

GARMENT WORKER PRODUCTIVITY PREDICTION

TEAM:

SHREYAS REDDY GUVVALA

NIKHIL PULAGAM

MULE VIGNESH

1. Model Development Phase

    a. Feature Selection Report

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying seven regression algorithms. The best model is saved based on its performance.

Activity 1.1: Linear Regression Model

```
lr = LinearRegression()
lr.fit(X_train,y_train)
```

```
LinearRegression
LinearRegression()
```

This code performs linear regression modeling using the scikit-learn library.

It creates a LinearRegression object named "lr". The "fit" method is then called on the training data X_train and y_train. This method trains the model by finding the coefficients for the linear equation that best fits the data.

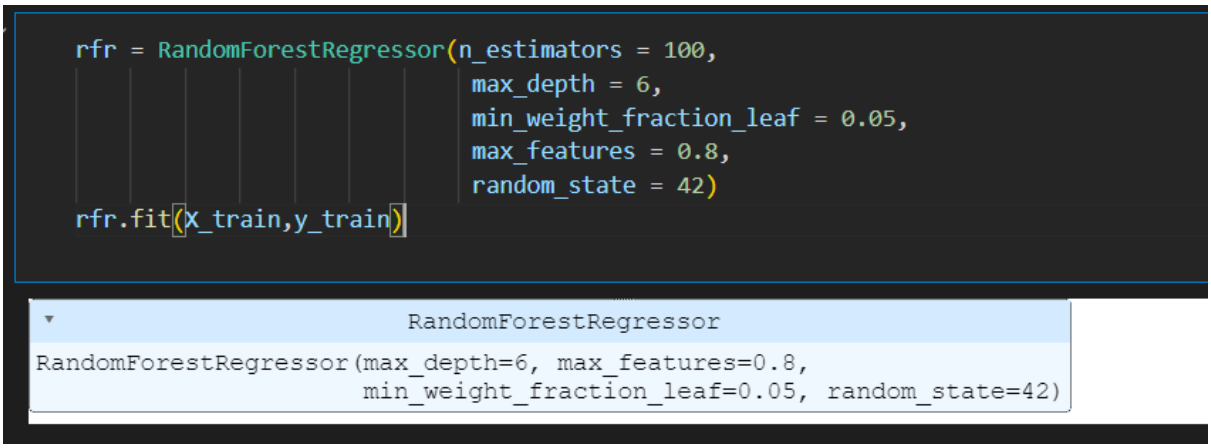
Activity 1.2: Decision Tree Regressor Model

```
dtr = DecisionTreeRegressor(max_depth= 4, min_samples_split= 3, min_samples_leaf= 2)
dtr.fit(X_train,y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=4, min_samples_leaf=2, min_samples_split=3)
```

This code is training a decision tree regression model using the training data (X_train and y_train) with the specified hyperparameters (max_depth=4, min_samples_split=3, min_samples_leaf=2). The model is stored in the variable 'dtr'. After training, the model will be able to make predictions on new data based on the relationships learned from the training data.

Activity 1.3: Random Forest Regressor Model



The code creates an instance of the RandomForestRegressor class with certain hyperparameters set. The hyperparameters specify the number of trees to use in the random forest, the maximum depth of each tree, the minimum weight fraction required to be at a leaf node, the maximum number of features to consider when splitting a node, and a random state to ensure reproducibility of results. The rfr object is then trained on the training data X_train and y_train using the fit method.

The trained model can then be used to make predictions on new data.

4.2Model Selection Report

Activity 1.4: Gradient Boosting Regressor Model

```
gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=1, random_state=42)
gbr.fit(X_train,y_train)
```

```
GradientBoostingRegressor
GradientBoostingRegressor(max_depth=1, random_state=42)
```

The code is fitting a Gradient Boosting Regressor model to the training data X_train and y_train using n_estimators equal to 100 (the number of trees in the forest), learning_rate equal to 0.1 (the step size shrinkage used to prevent overfitting), max_depth equal to 1 (the maximum depth of the individual regression estimators), and random_state equal to 42 (to ensure reproducibility of results).

The fitted model can then be used to make predictions on new data using the predict() method.

Activity 1.5: Extreme Gradient Boost Regressor Model

```
xgb = XGBRegressor(n_estimators=300, learning_rate=0.05,
                   max_leaves = 3, random_state = 1)
xgb.fit(X_train, y_train)
```

This code is training an XGBoost regression model. The XGBRegressor function is being used to create the model. The parameters passed to the function include the number of estimators, learning rate, maximum number of leaves, and random state. The model is trained on the training set (X_train and y_train) using the fit() method. Once trained, the model can be used to make predictions on new data.

4.3 Initial Model Training Code, Model Validation and Evaluation Report

Activity 1.6: Bagging Regressor Model

```
# Define base model
base_model = XGBRegressor(n_estimators=700, learning_rate=0.06, max_depth=2, max_leaves=3, random_state=1)

# Create bagging regressor
bagging_reg = BaggingRegressor(base_model, n_estimators=100, random_state=42)


# Fit bagging regressor
bagging_reg.fit(X_train, y_train)
```

This code creates a machine learning model using the XGBoost algorithm. The model is designed to predict a numeric value (the target variable) based on a set of input variables.

To improve the accuracy of the model, the Bagging technique is used. This involves creating multiple versions of the model, each with slightly different training data, and combining their predictions to create a final prediction.

Finally, the model is trained using a set of input data (X_train) and the corresponding target values (y_train). This involves adjusting the weights of the various components of the model until it can accurately predict the target values based on the input data.

Activity 1.7: Boosting Regressor Model

```
# Define base model
base_model = XGBRegressor(n_estimators=700, learning_rate=0.06, max_depth=2, max_leaves=3, random_state=1)

# Create AdaBoost regressor
boosting_reg = AdaBoostRegressor(base_model, n_estimators=100, learning_rate=0.1, random_state=42)

# Fit AdaBoost regressor
boosting_reg.fit(X_train, y_train)
```

This code creates a machine learning model to predict a numeric value based on a set of input variables using the XGBoost algorithm as the base model.

To improve the accuracy of the model, the AdaBoost technique is used. AdaBoost stands for Adaptive Boosting and works by creating multiple versions of the model, each with slightly different training data, and weighting the predictions of each model based on its accuracy.

Finally, the model is trained using a set of input data (X_train) and the corresponding target values (y_train). This involves adjusting the weights of the various components of the model until it can accurately predict the target values based on the input data.