Graduate School of Engineering and Science

CS 552 – Data Science with Python

2020 FALL

Instructor : Dr. Reyhan AYDOĞAN

Assignment 3 : Image Classification & Dimensionality Reduction

Salih Emre KAYNAR – S011496

1 . Introduction

This assignment demonstrates solving of Image Classification and Dimensionality Reduction problem by several Classification techniques and PCA Dimensionality Reduction technique. These techniques are applied on Fashion MNIST provided by Zalando Research. Fashion MNIST is basically MNIST like fashion product dataset. Jupyter IPython Notebook is used for providing environment for this problem.

5 Classification techniques are implemented in this assignment. These techniques are : K Nearest Neighbors, Decision Tree, Perceptron, Random Forest and Support Vector Machine. All these techniques are based on Supervised Machine Learning Technique

2 . Methodology

Supervised Machine Learning

Supervised Machine Learning technique is mostly used to solve classification and regression problems. This technique is basically making predictions by a given input data. With the given labeled input data, the algorithm builds a function and makes a prediction about the label of given new unlabeled data.

K-Nearest Neighbors

K-Nearest neighbors(KNN) is a supervised machine learning technique which can be used for Classification problems. This technique assumes that similar features exist in the dataset and similar data points are close to each other. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification).[1]

Decision Tree

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions are performed on the features of the given dataset. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. A decision tree simply asks a question and it further split the tree into subtrees based on the answer.

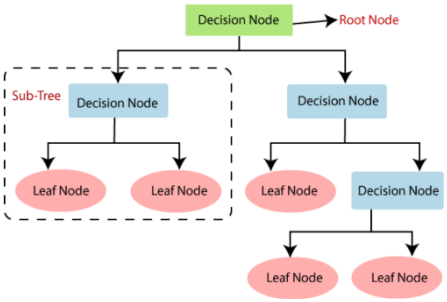

Figure 1. Decision Tree Diagram

Perceptron

Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample[2].

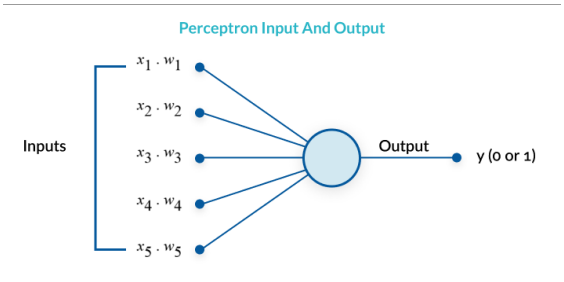

Figure 2. Perceptron Input and Output

Random Forest

Random forest can be considered as an ensemble of decision trees. As training process of Random Forest, multiple decision trees are growing, and the prediction is made by each tree to assign the class label by majority vote. Random forest corrects for decision trees' habit of overfitting to their training set.

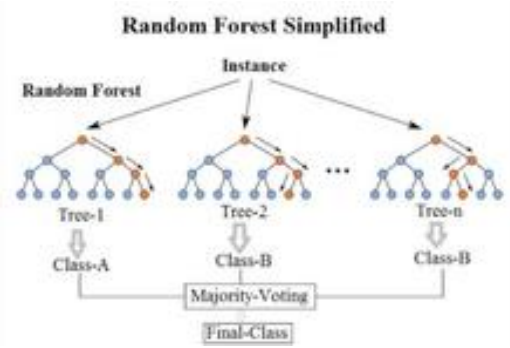

Figure 3. Random Forest Diagram

Support Vector Machine

In the Support Vector Machine(SVM), the algorithm creates a line or a hyperplane which separates the data into classes. Support vectors are the data points that lie closest to the hyperplane. he decision boundary that maximizes the distance from the nearest data points of all the classes. The decision boundary created by SVMs is called the maximum margin classifier or the maximum margin hyper plane.[3]

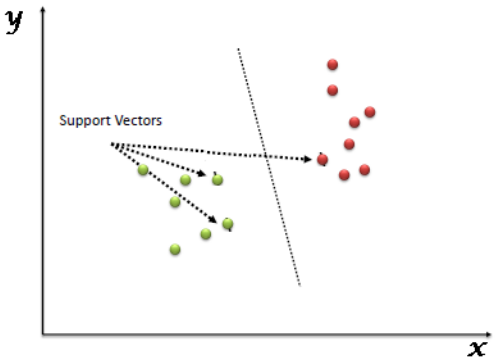

Figure 4. Support Vectors

PCA

Principle Component Analysis is explaining a multi-variable numerical dataset with less variables and minimum information loss. Basically, reducing dimension inside a dataset. PCA does this by finding eigen vectors and eigen values of covariance matrix. After finding these values number of principal components are selected by eliminating relatively unimportant eigen vector. Then eigen values are sorted and paired with opposing eigen vectors. The new reduced dataset is calculated by multiplication of eigen vectors transposed and the scaled input data.

Fashion MNIST Dataset

Fashion MNIST is a dataset which contains 28 x 28 grayscale images of 70.000 fashion products from 10 categories, with 7000 images per category. Training set of the dataset has 60.000 images while the test set has 10.000 images. The categories are : ‘T-shirt/top’, 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'. An example of 8 images from the dataset is shown below Figure 5.

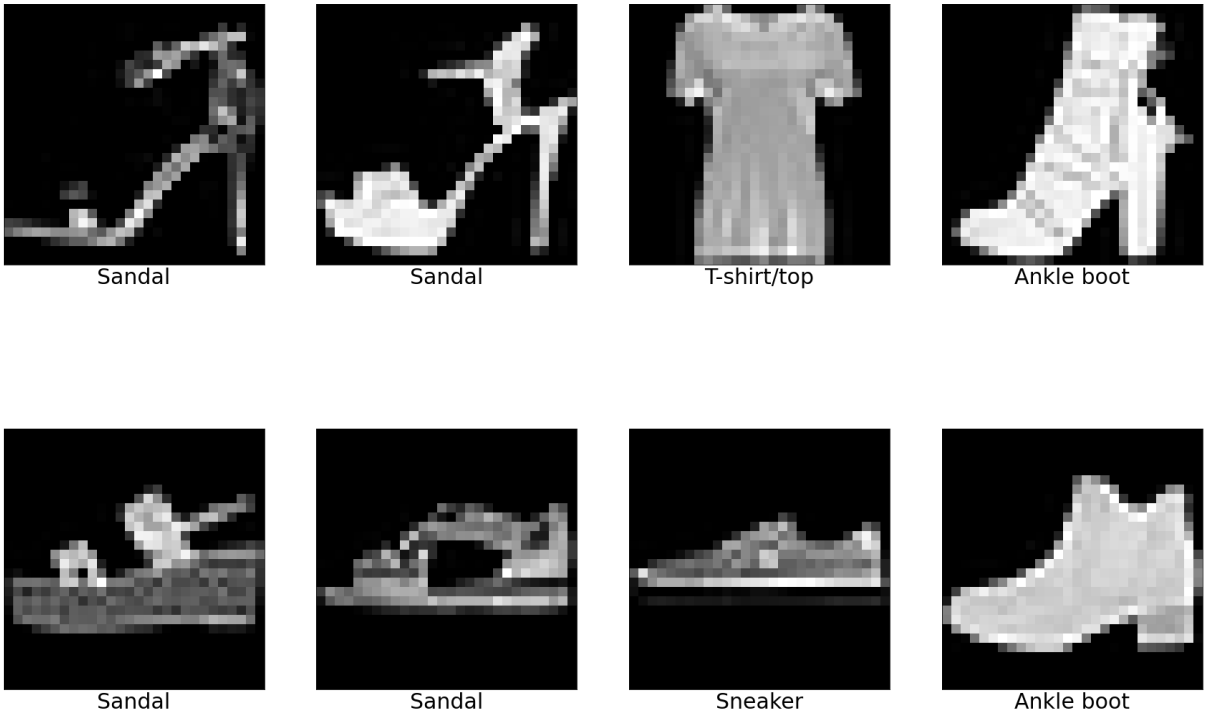

Figure 5: 8 images from Fashion MNIST dataset

3. Implementation Details

Data Analysis

The Fashion MNIST dataset is imported to the notebook by “load_mnist()” function from “mnist_reader” library provided by Zalando Research. The dataset is divided into 4 files : 60.000 Training Images, Labels of training images, 10.000 testing images, Labels of testing images. The images are imported in 2D numpy ndarray type. Each row represents an image while each column represents the color of each pixels between 0 to 255 for grayscale colors. The labels are imported in numerical form between 0 to 9.

Preprocessing

Before building our model, we need to process our data for a more precise model. In this assignment, scaling the features(pixels) between 0 to 1 in order build our model much faster and more precise.

Imported labels for images are in numeric form so a list of strings is defined where the indexes corresponding to each number of fashion product

Since each fashion image is in vector form, we need to convert it to a matrix in order to plot images with matplotlib library. We can simply do this by using numpy.reshape() function.

Model Training

The models are trained with the scaled training data and the corresponding labels and predicted with the scaled test data.

For K Nearest Neighbors, number of neighbors are selected as 5 and “minkowski” distance metric are used as hyperparameters.

For Decision Tree, criterion is selected as entropy. The splitter is selected as “best” in order to choose the best split automatically by the algorithm.

For Perceptron maximum of 50 iterations is selected for performance

For Random Forest Classifier the same criterion “entropy” is used in order to match with Decision Tree.

For Support Vector Classification, 1 for C value and polynomial kernel is chosen because our date is obviously non-linear.

PCA

4 different explained variance ratios are applied for each model(%25, %50, %75, %95). Number of components for each desired ratios are calculated by a basic for loop. The loop sums the sorted explained variance ratios and counts the number of iterations until it reaches the desired value. After the number of components are calculated, the PCA models are applied for each training and test data

Evaluation Metrics

In order to evaluate our predictions, several evaluation metrics are used to measure the performance of our models. First Evaluation method is calculating the accuracy rate. In addition to that I also plot 12 randomly chosen predicted labels and its true labels with the corresponding image. For the last metric, I constructed the confusion matrix with the method from sklearn.metrics and plot with a heatmap from Seaborn library. Finally, all of these results are gathered to a dataframe with pandas library.

4. Results & Conclusion

Results are based on the output of evaluation metrics functions. The summarized accuracy results are show below in Table 1.

	Classification Method	Accuracy	Number of Components	Ratio of Explained Variance
0	K Nearest Neighbours	0.2424	1	25
1	K Nearest Neighbours	0.6234	3	50
2	K Nearest Neighbours	0.8274	14	75
3	K Nearest Neighbours	0.8622	187	95
4	K Nearest Neighbours	0.8554	784	100
5	Decision Trees	0.3078	1	25
6	Decision Trees	0.6327	3	50
7	Decision Trees	0.7679	14	75
8	Decision Trees	0.7800	187	95
9	Decision Trees	0.8120	784	100
10	Perceptron	0.1363	1	25
11	Perceptron	0.4452	3	50
12	Perceptron	0.7048	14	75
13	Perceptron	0.7909	187	95
14	Perceptron	0.8142	784	100
15	Random Forests	0.2240	1	25
16	Random Forests	0.6383	3	50
17	Random Forests	0.8422	14	75
18	Random Forests	0.8580	187	95
19	Random Forests	0.8796	784	100
20	SVC	0.2825	1	25
21	SVC	0.6255	3	50
22	SVC	0.8498	14	75
23	SVC	0.8963	187	95
24	SVC	0.8808	784	100

Table 1. Performance Summary of all models

Based on Table 1, explained ratios 25% and 50% got poor results with a very low number of components in all prediction models. After %75 we can see that we're getting more than 80% accuracy in most of the models.

SVC with %95 explained variance has the highest accuracy rate among all models with accuracy rate of %89.6. Number of components are more than 4 times reduced, which means higher accuracy with less computational time and less memory. PCA dimensional reduction is functional in this model. On the other hand, we can say that SVC and KNN is likely for overfitting when fitting the model with all components.

Perceptron and Decision Tree models are not useful approaches for this problem as we can see above with the lowest accuracy rates.

In conclusion, the results and models showed me that we can use machine learning for finding classes of images with a classified training image dataset. In addition to that, PCA dimension reduction technique may be useful for faster and more accurate predictions.

Appendix – Evaluation Metric for each model

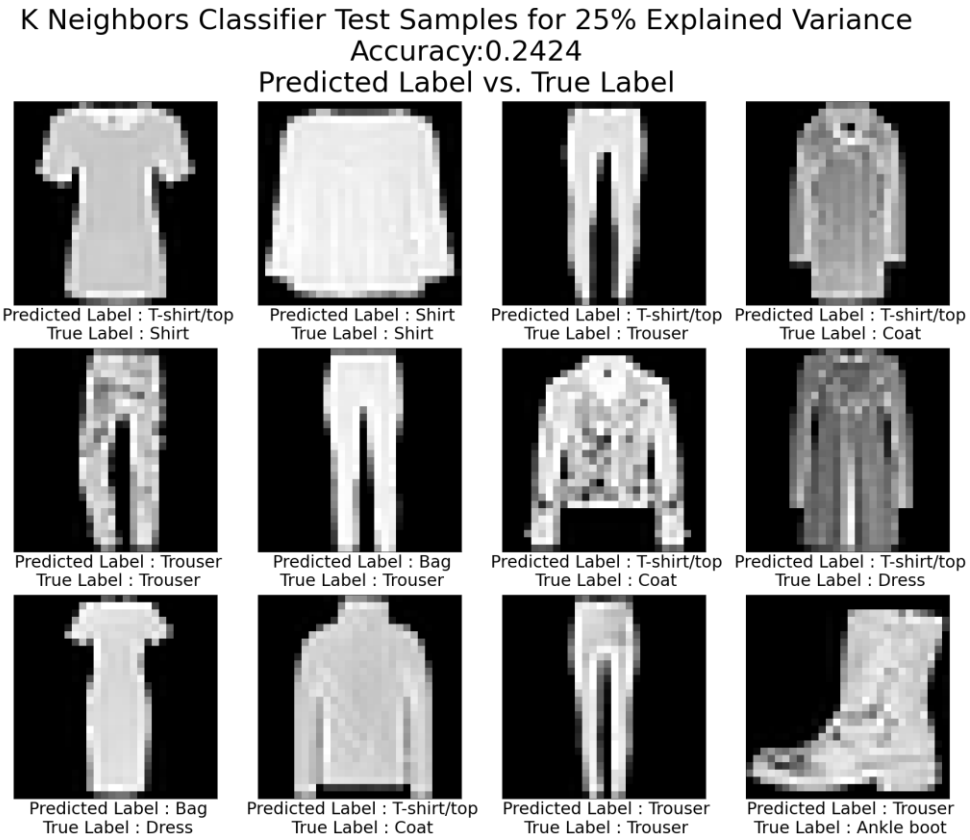

Figure 6. KNN sample predictions for 12 images for 25% explained variance

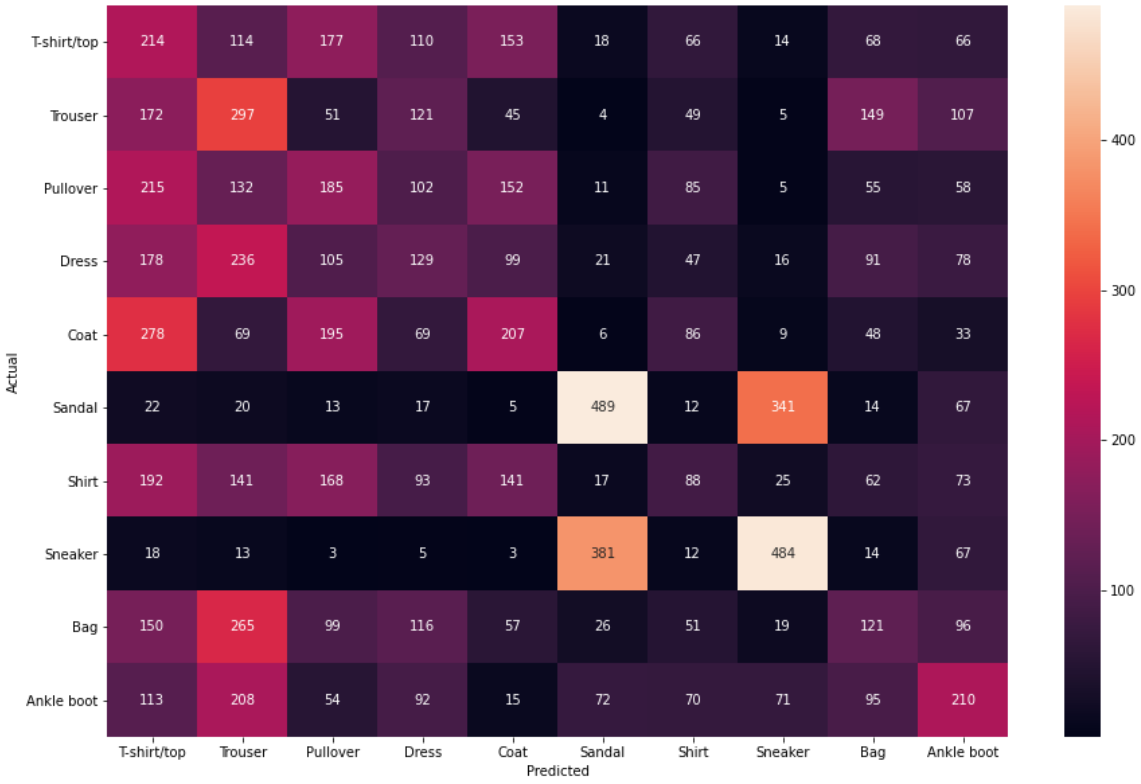

Table 2. Confusion Matrix for KNN model with 25% explained variance

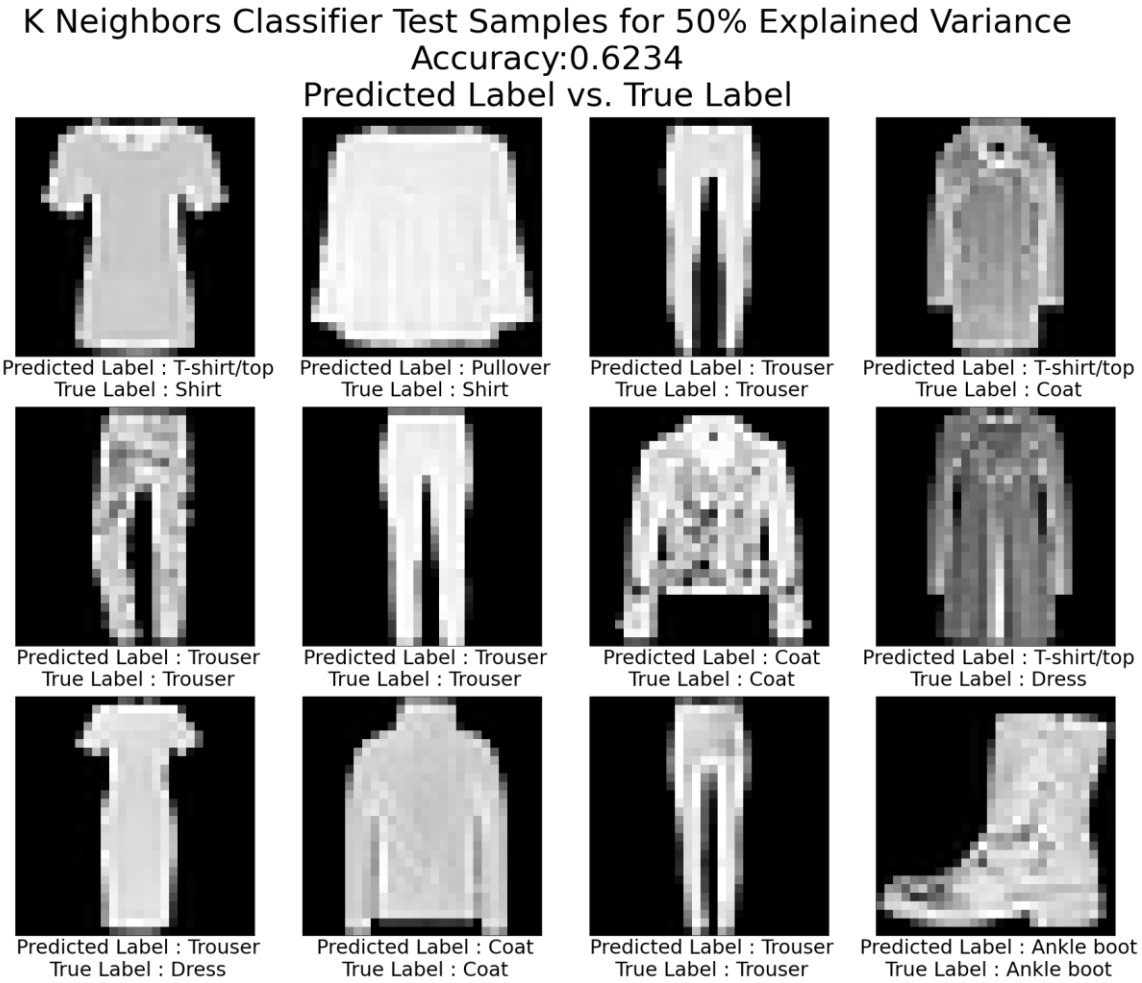

Figure 7. KNN sample predictions for 12 images for 50% explained variance

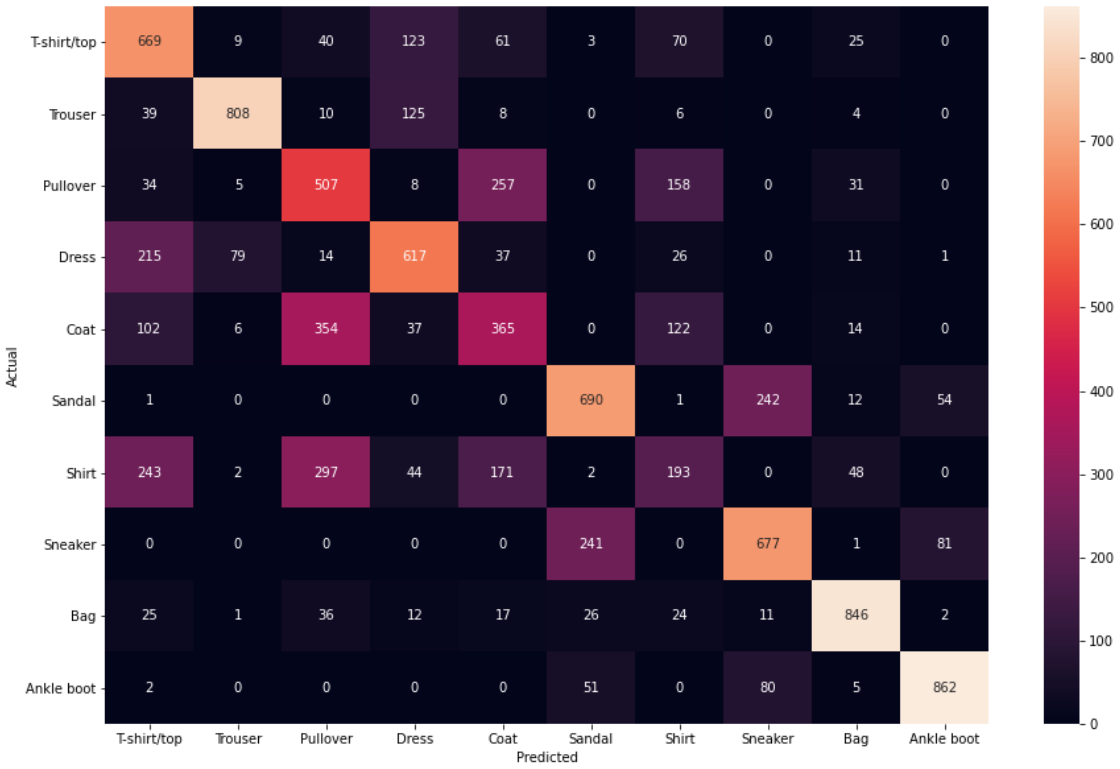

Table 3. Confusion Matrix for KNN model with 50% explained variance

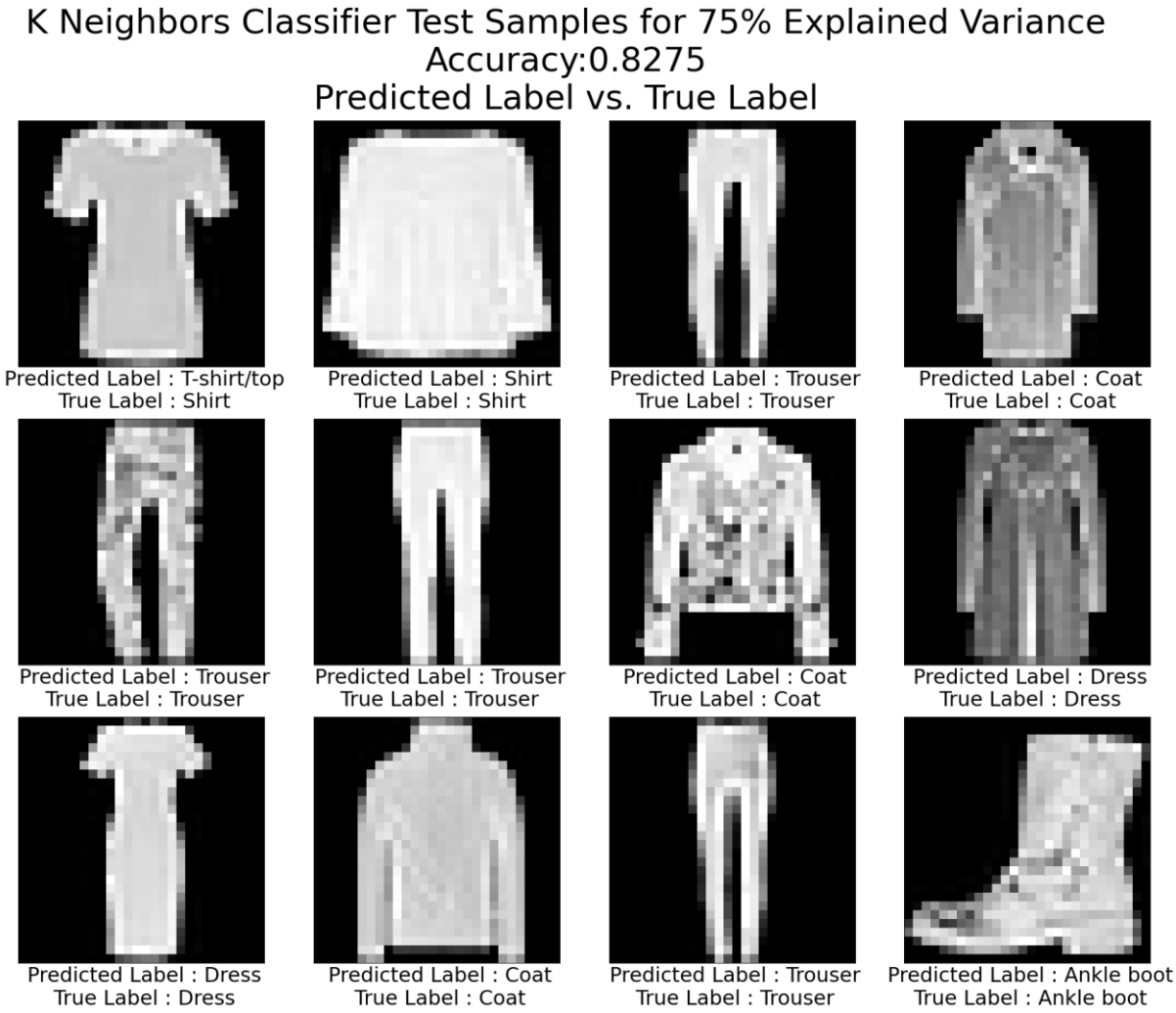

Figure 8. KNN sample predictions for 12 images for 75% explained variance

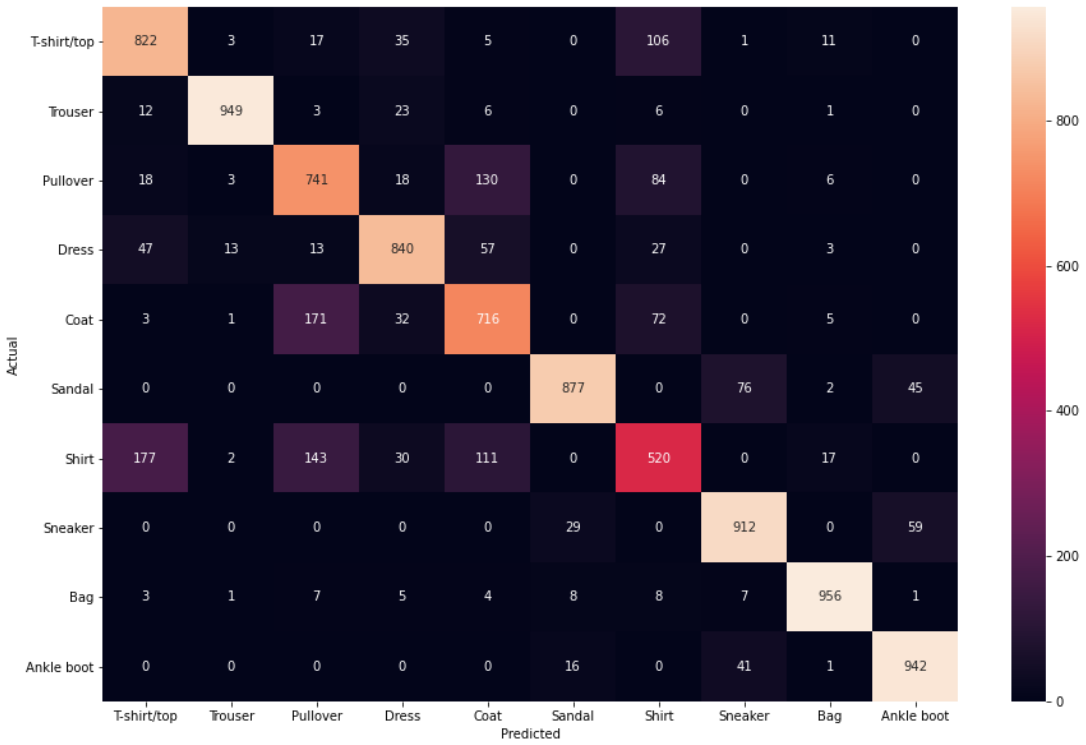

Table 4. Confusion Matrix for KNN model with 75% explained variance

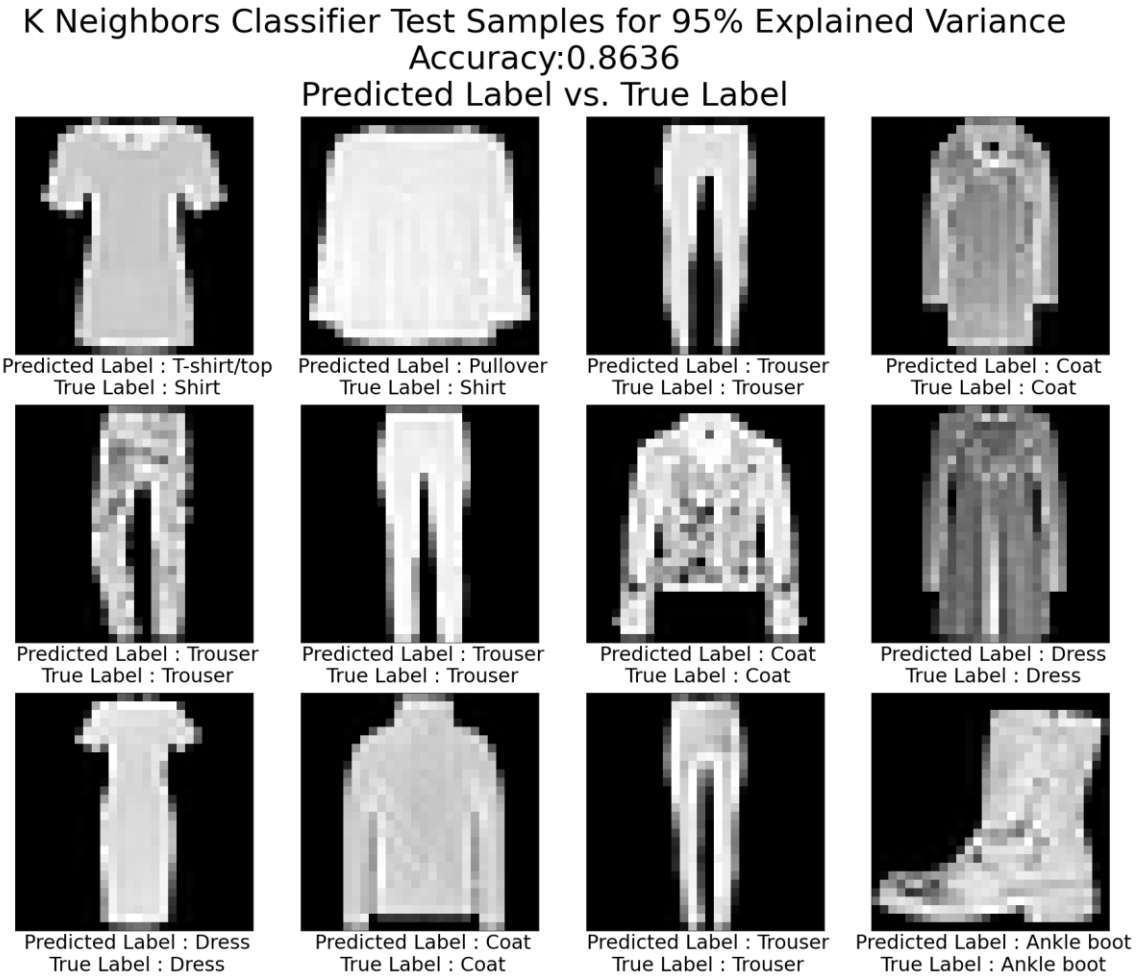

Figure 9. KNN sample predictions for 12 images for 95% explained variance

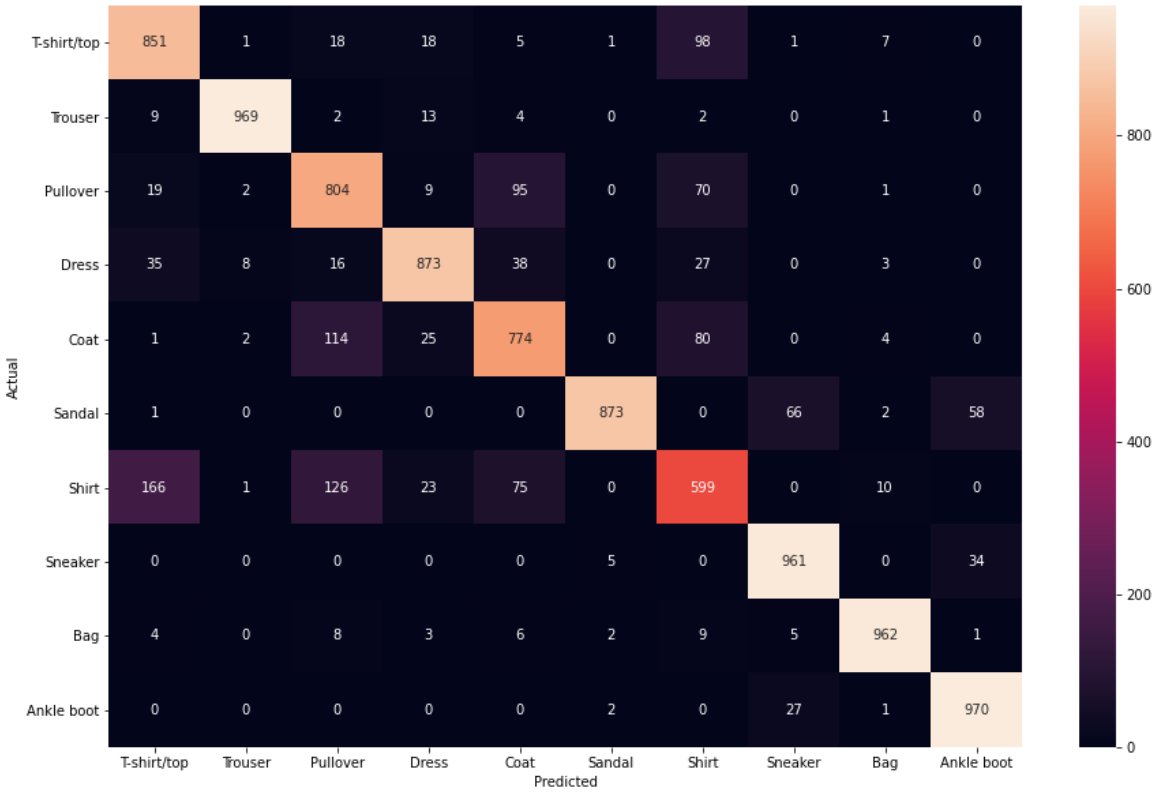

Table 5. Confusion Matrix for KNN model with 95% explained variance

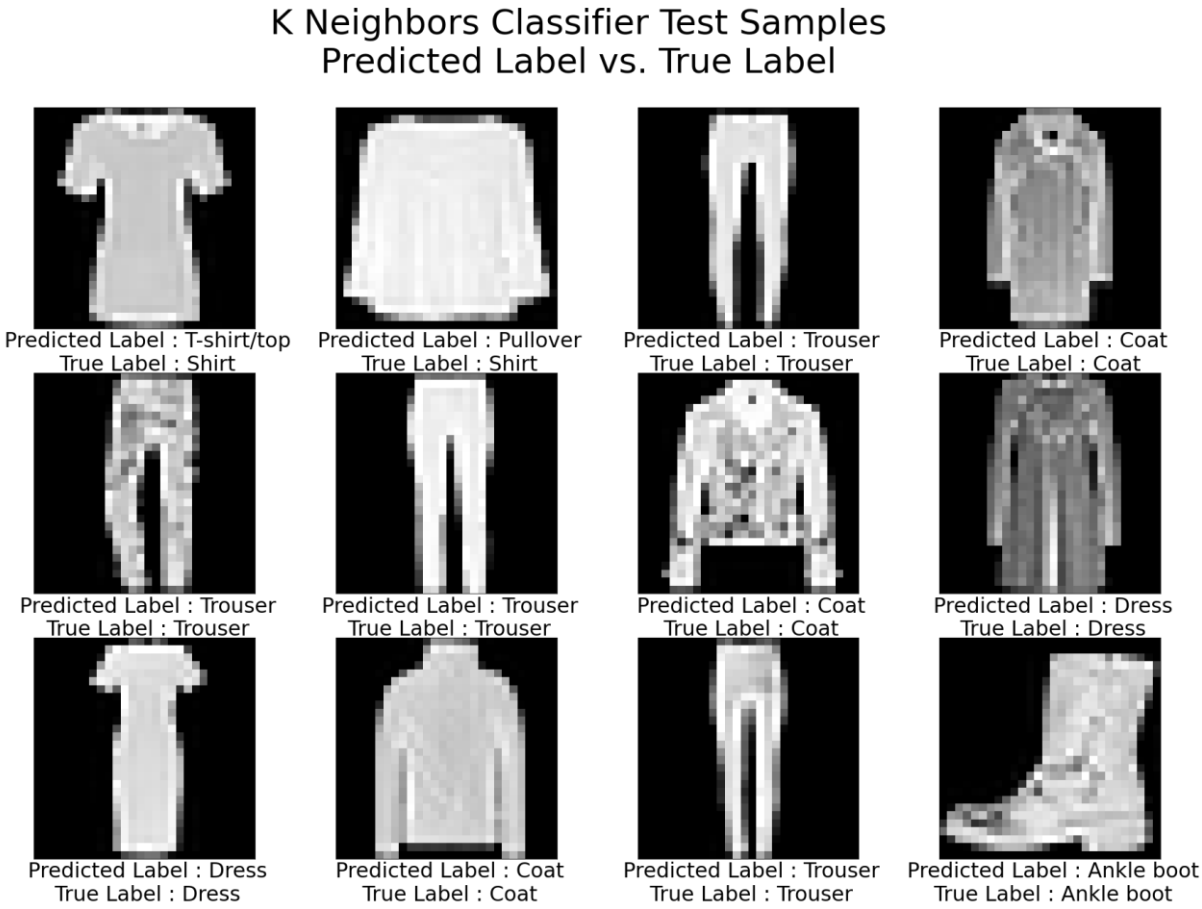

Figure 10. KNN sample predictions for 12 images for 100% explained variance

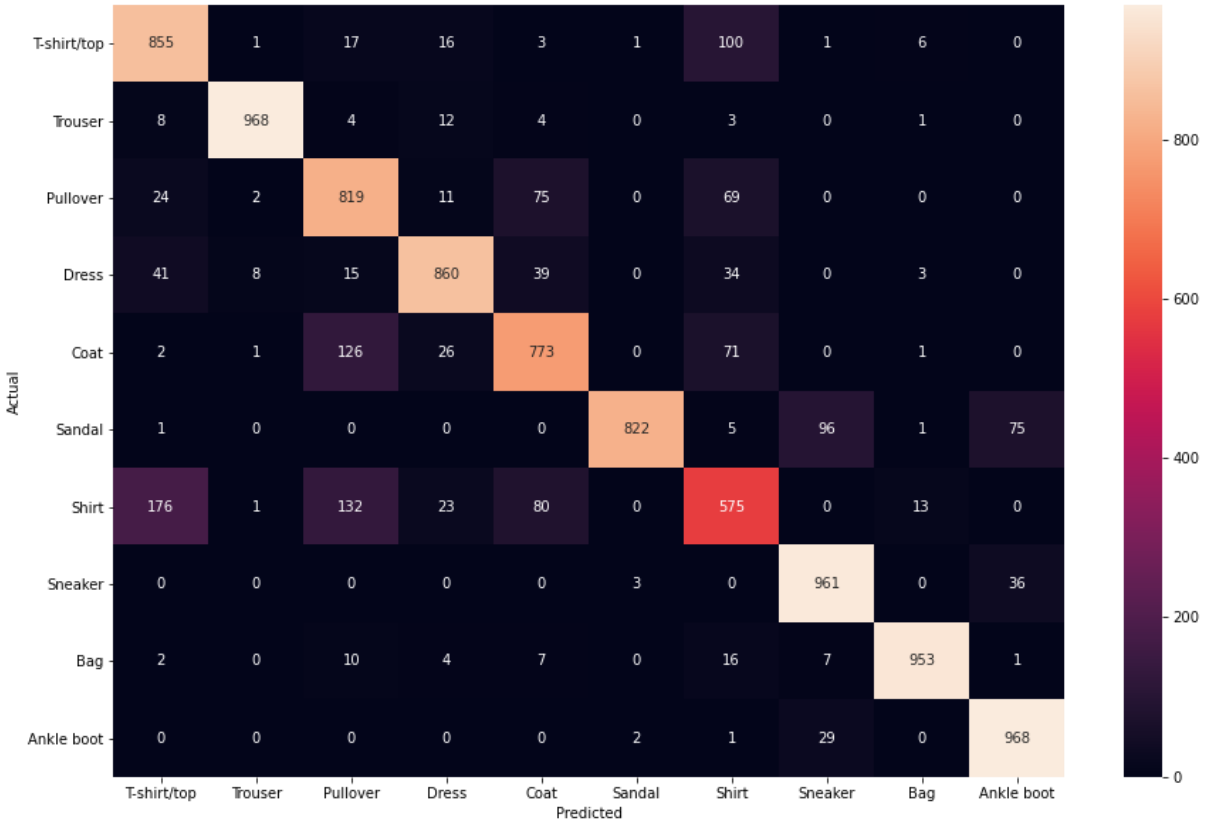

Table 6. Confusion Matrix for KNN model with 100% explained variance

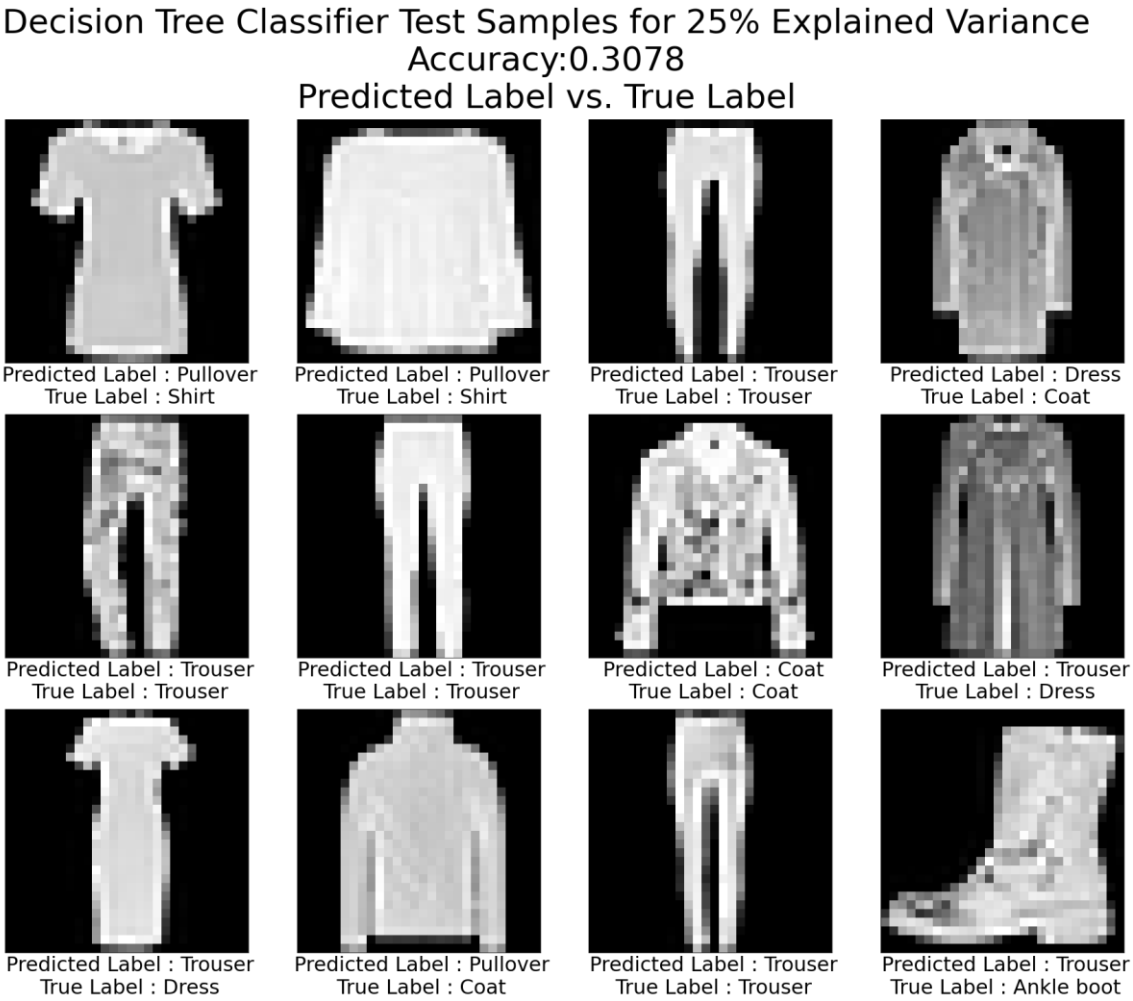

Figure 11. Decision Tree sample predictions for 12 images for 25% explained variance

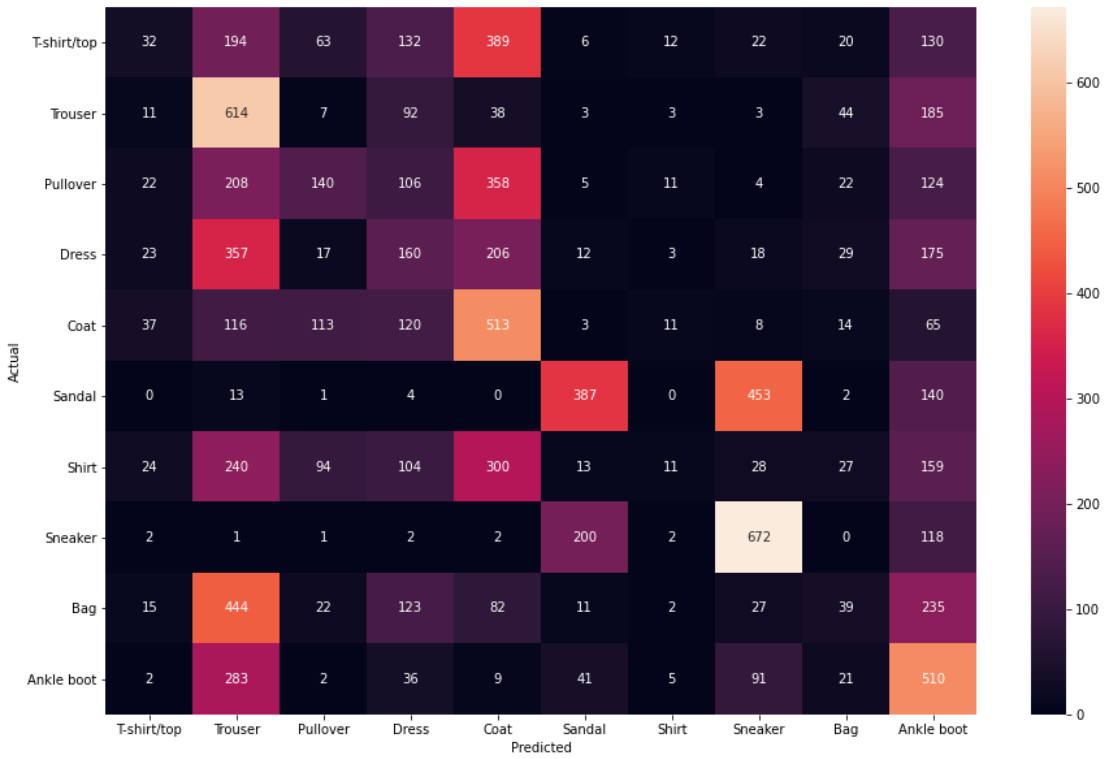

Table 7. Confusion Matrix for Decision Tree model with 25% explained variance

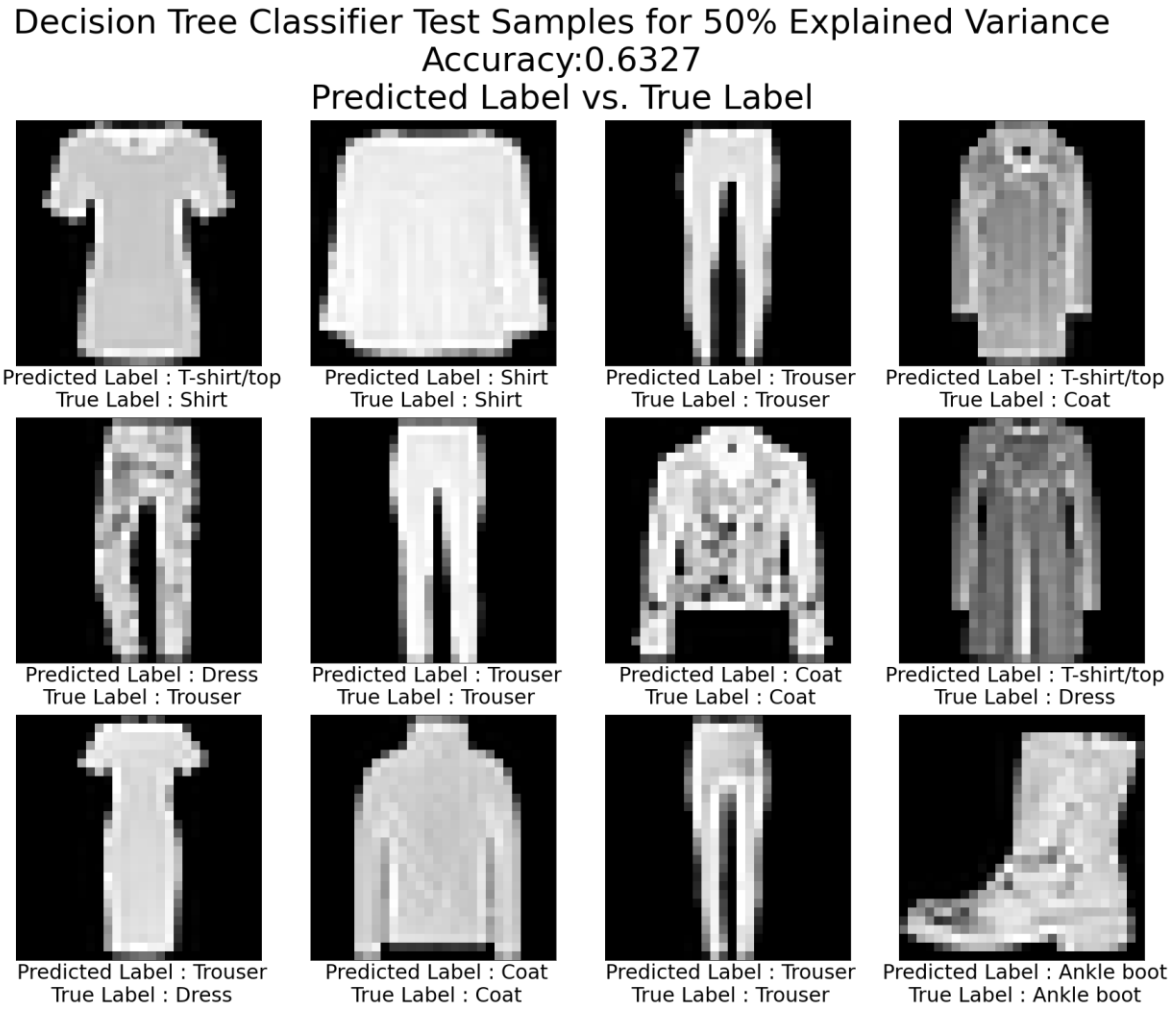

Figure 12. Decision Tree sample predictions for 12 images for 50% explained variance

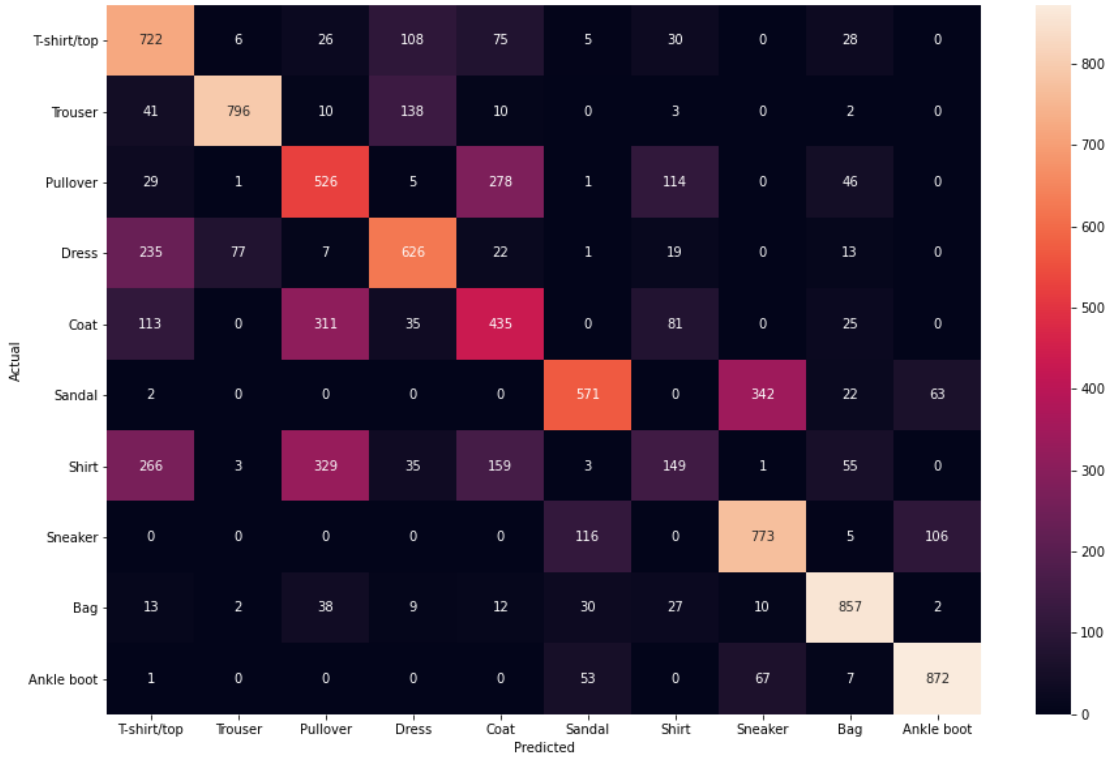

Table 8. Confusion Matrix for Decision Tree model with 50% explained variance

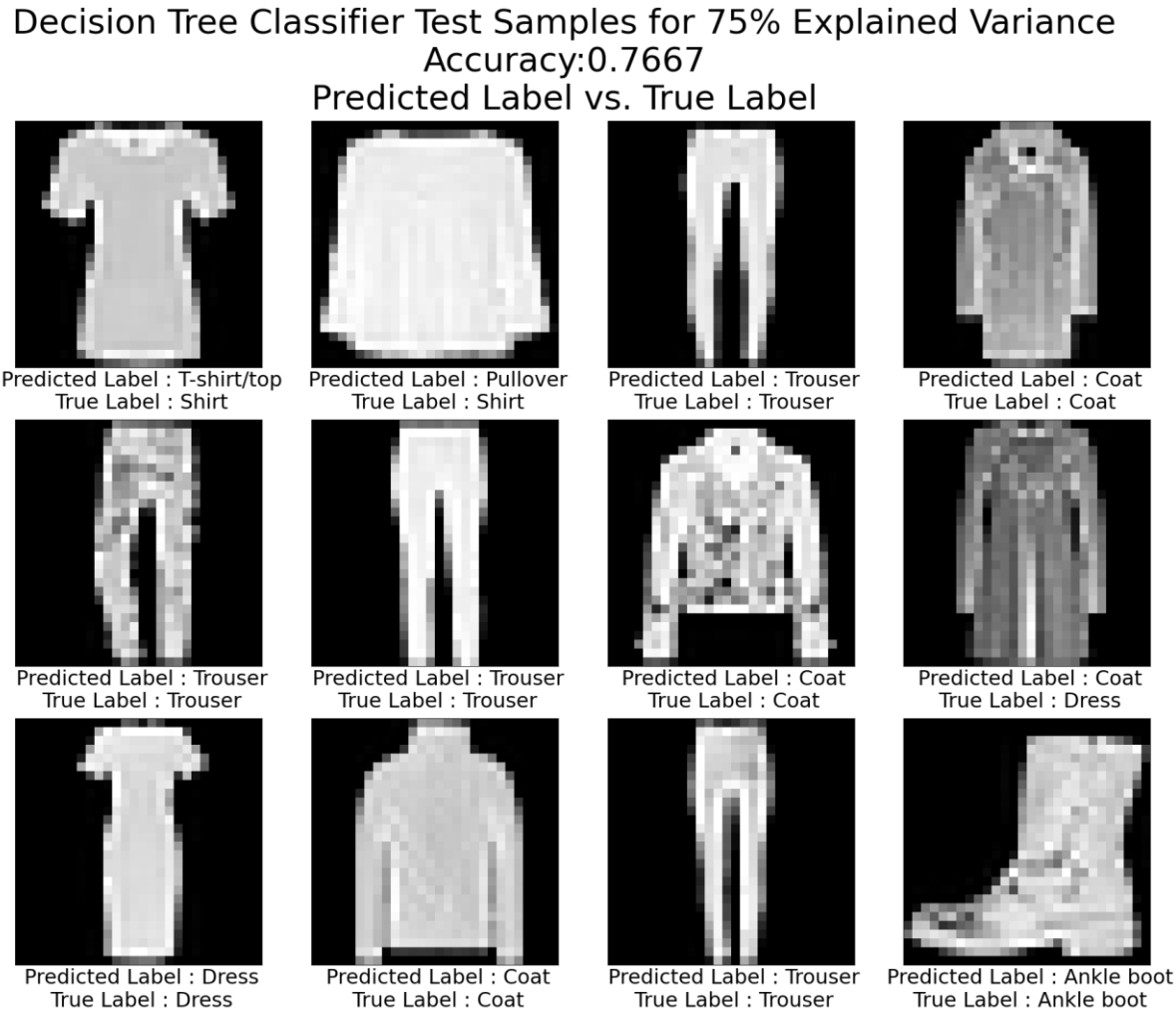

Figure 13. Decision Tree sample predictions for 12 images for 75% explained variance

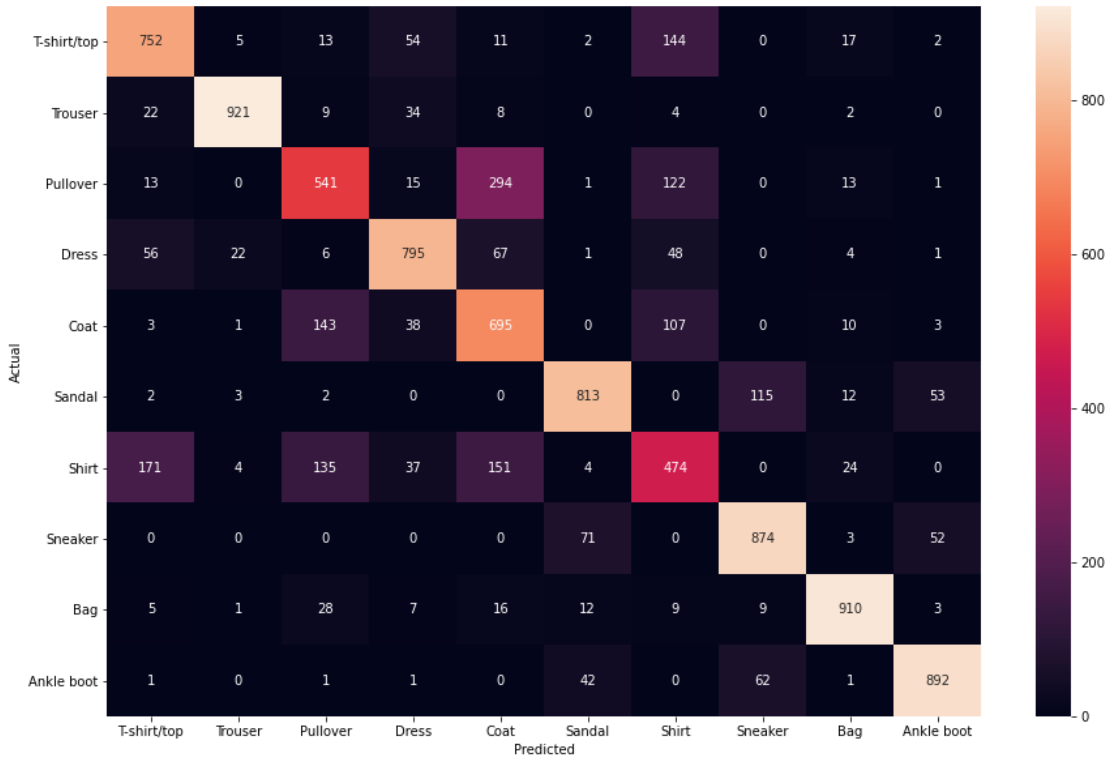

Table 9. Confusion Matrix for Decision Tree model with 75% explained variance

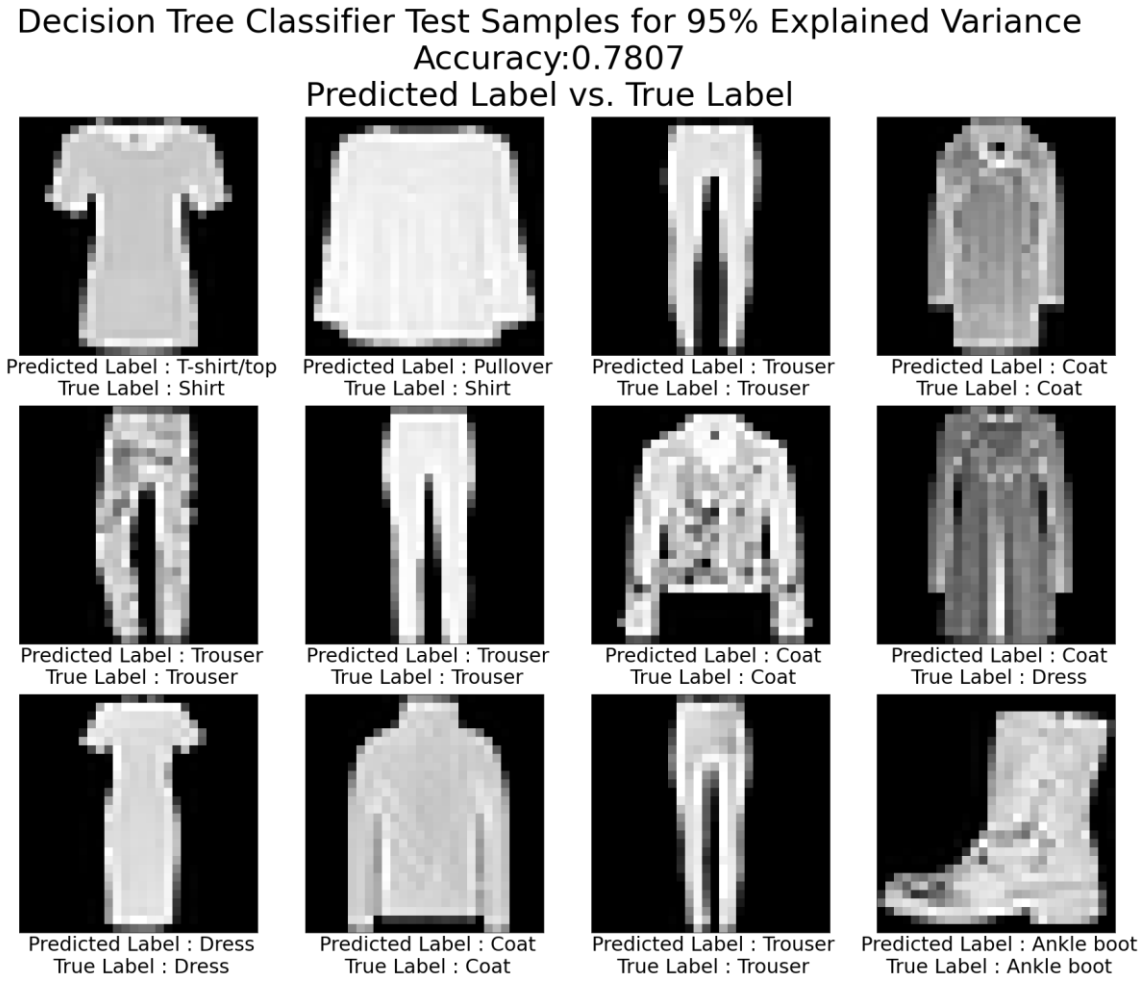

Figure 14. Decision Tree sample predictions for 12 images for 95% explained variance

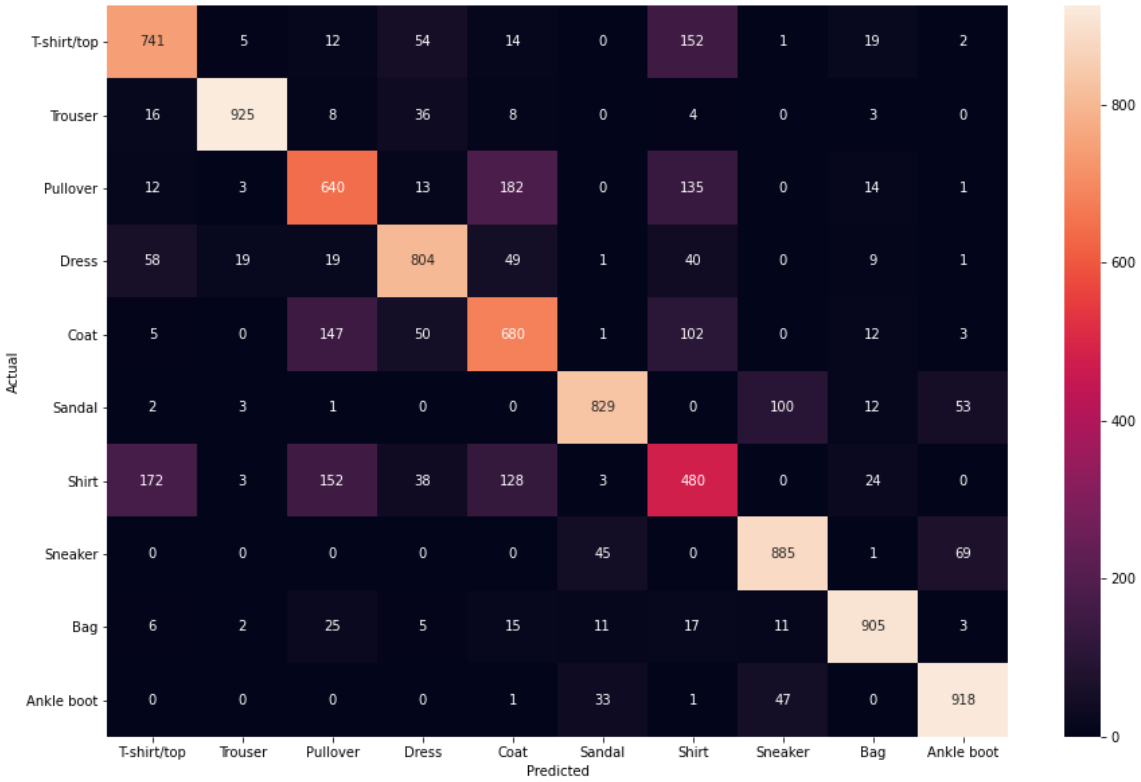

Table 10. Confusion Matrix for Decision Tree model with 95% explained variance

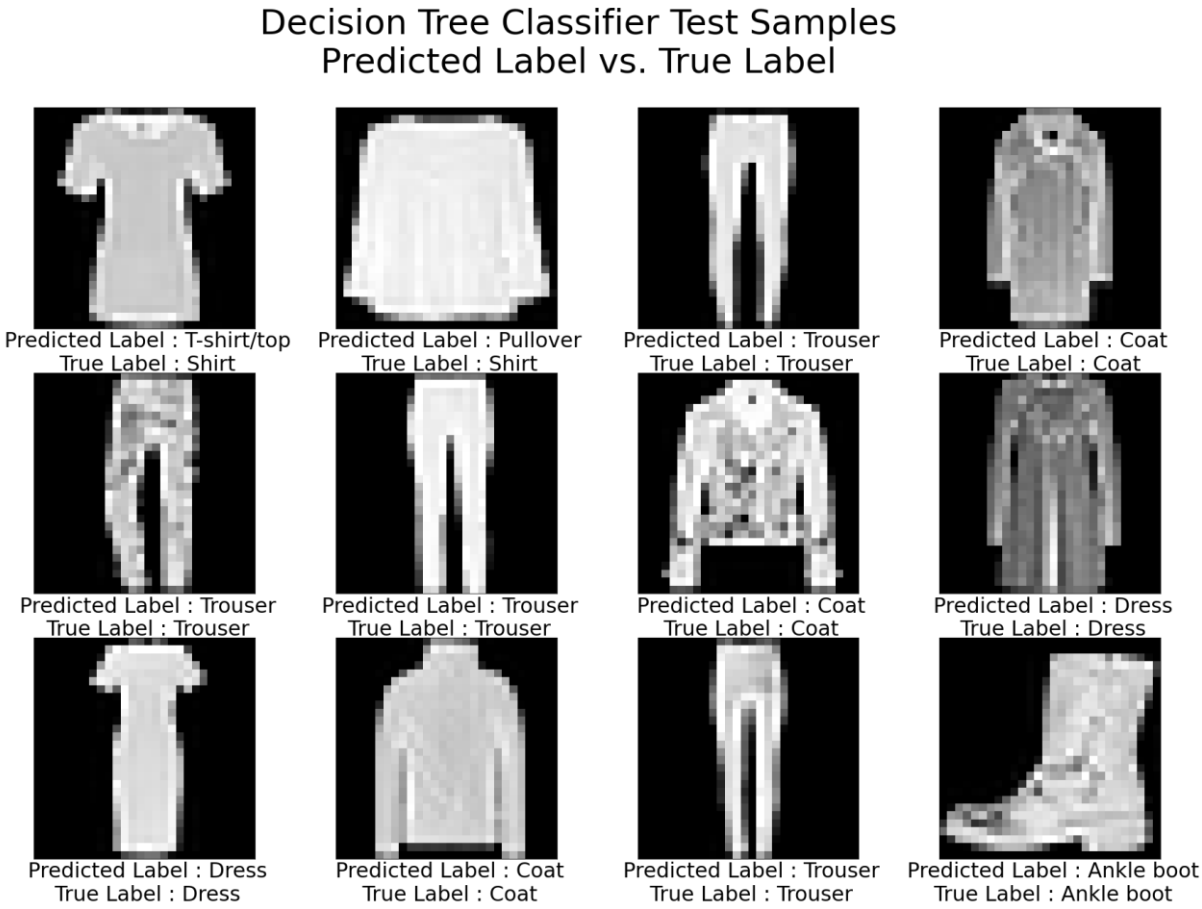

Figure 15. Decision Tree sample predictions for 12 images for 100% explained variance

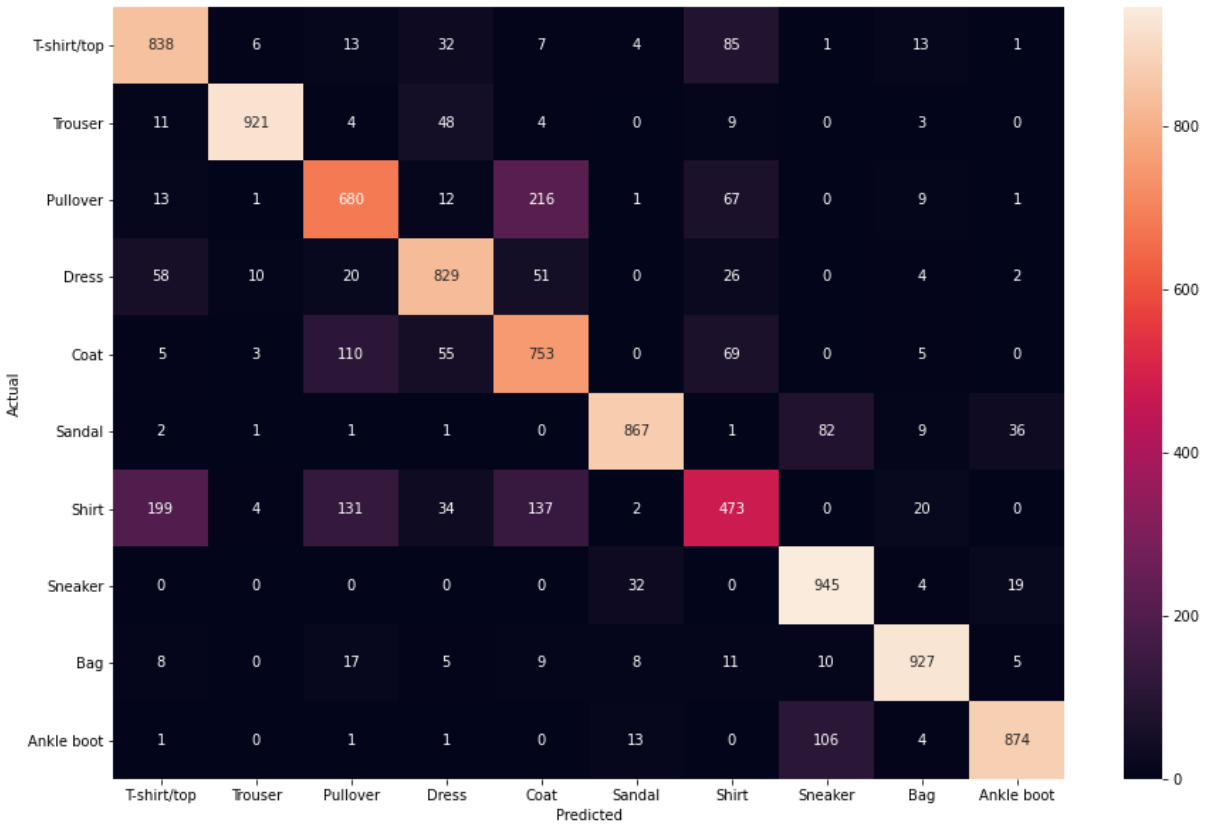

Table 11. Confusion Matrix for Decision Tree model with 100% explained variance

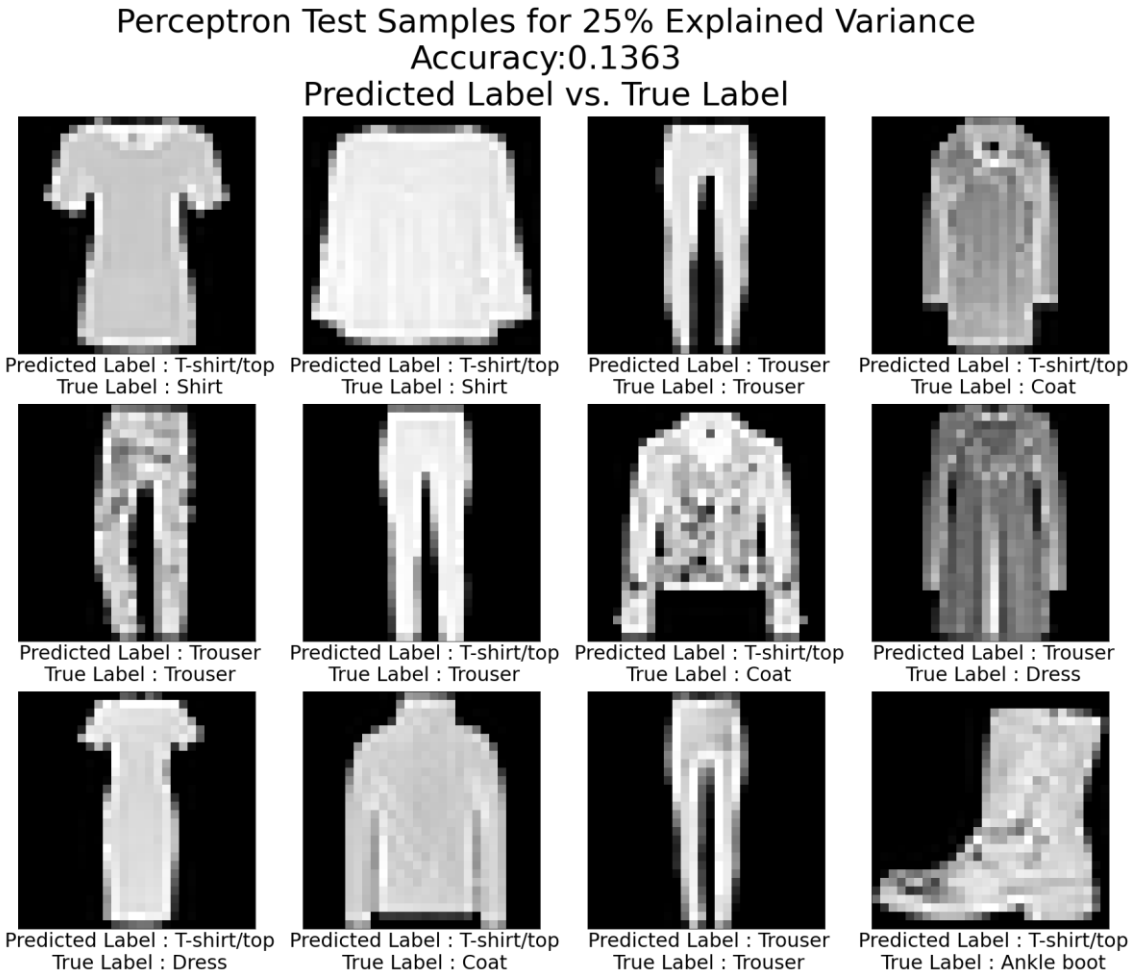

Figure 16. Perceptron sample predictions for 12 images for 25% explained variance

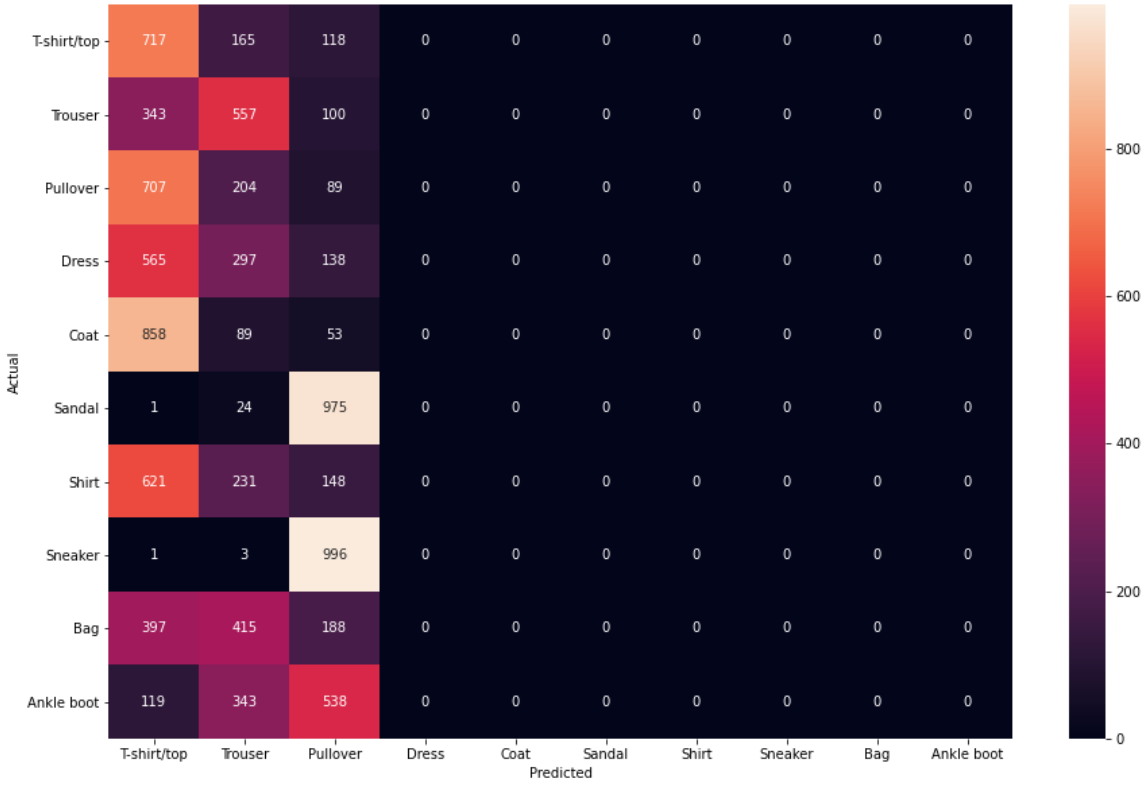

Table 12. Confusion Matrix for Perceptron model with 25% explained variance

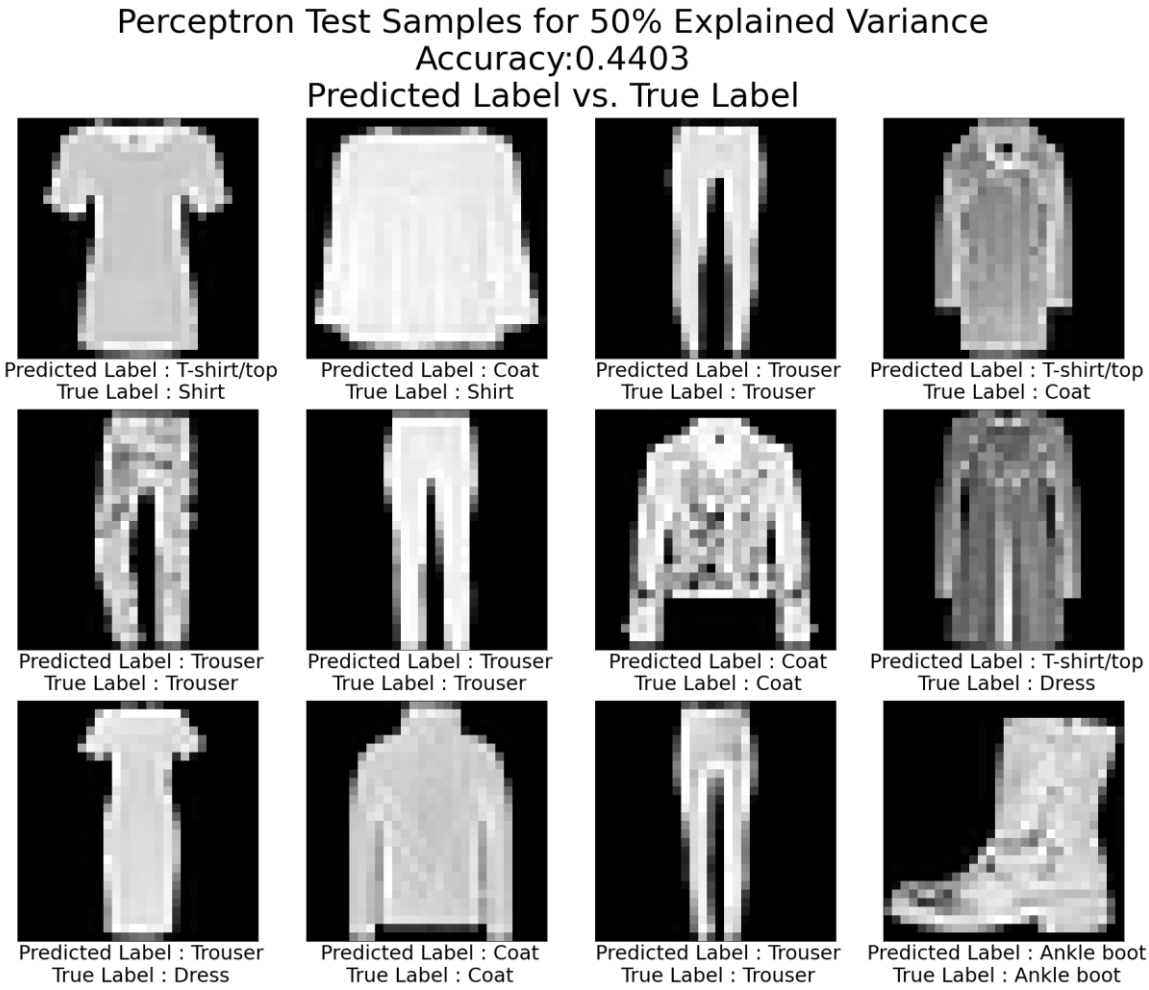

Figure 17. Perceptron sample predictions for 12 images for 50% explained variance

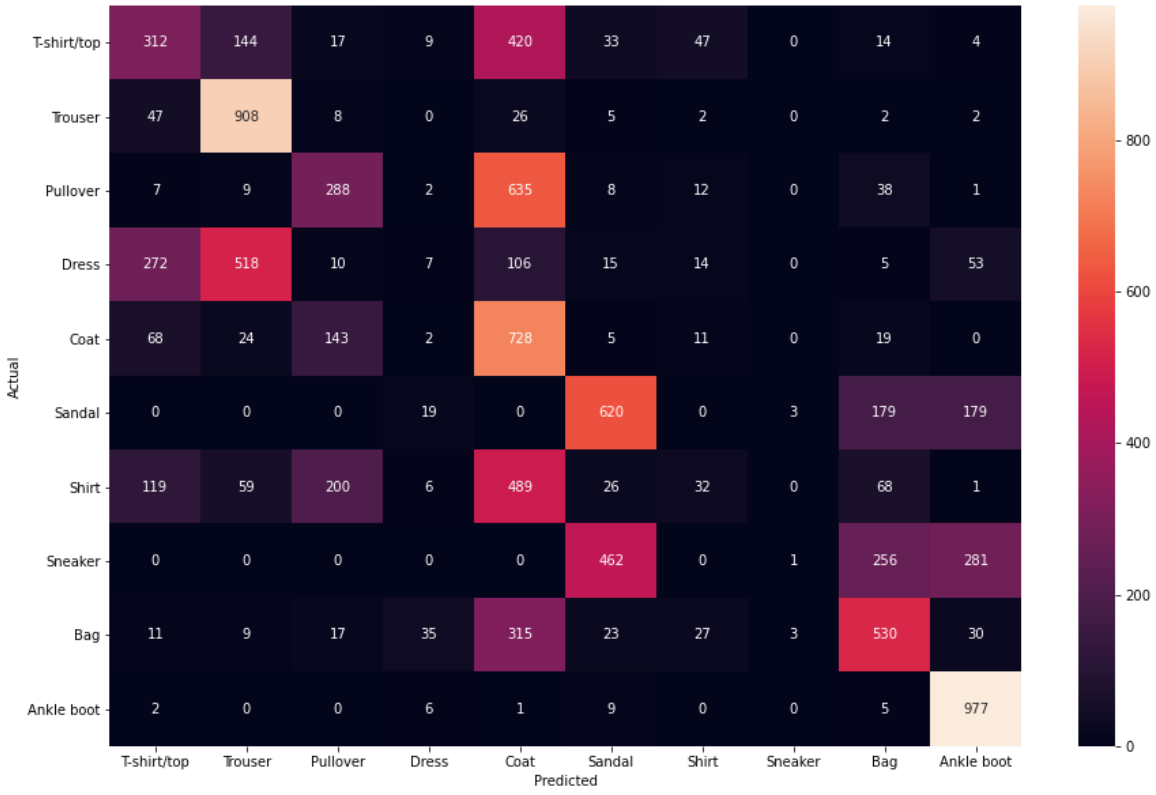

Table 13. Confusion Matrix for Perceptron model with 50% explained variance

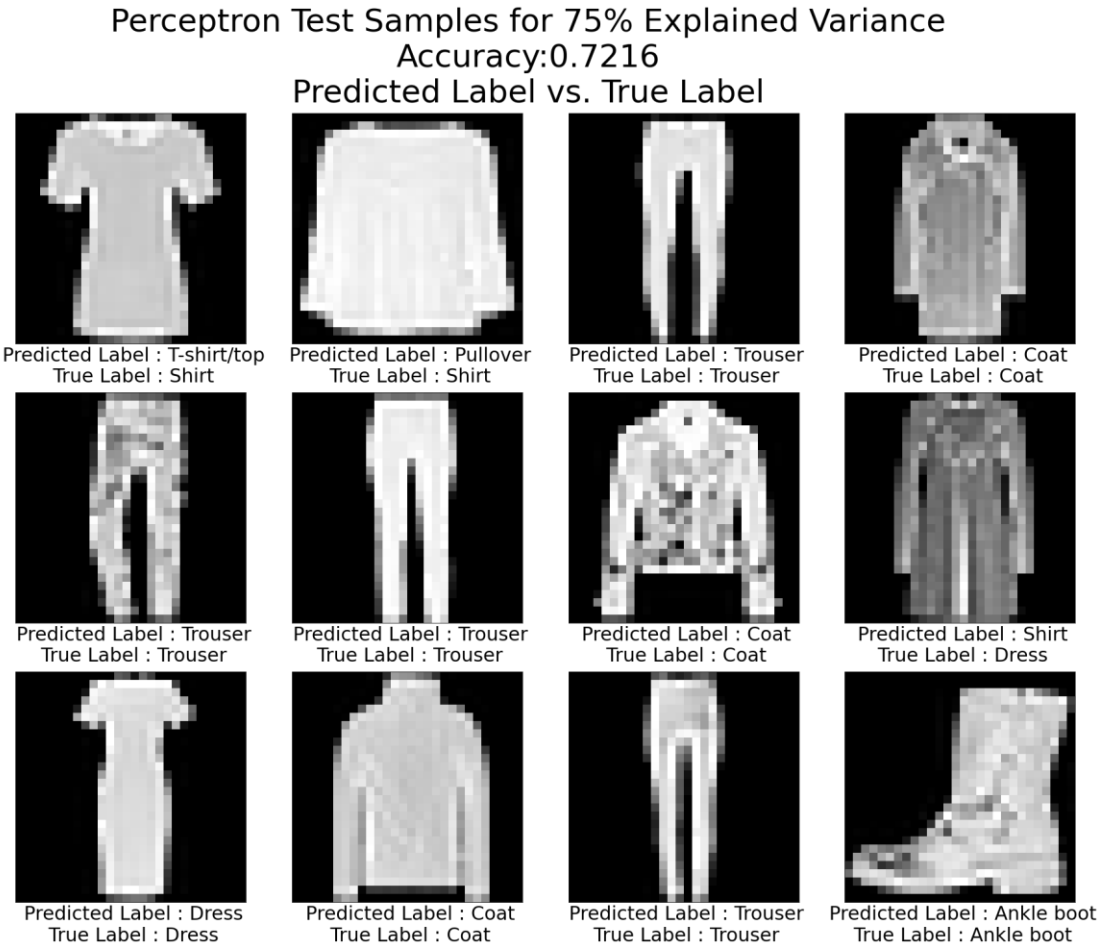

Figure 18. Perceptron sample predictions for 12 images for 75% explained variance

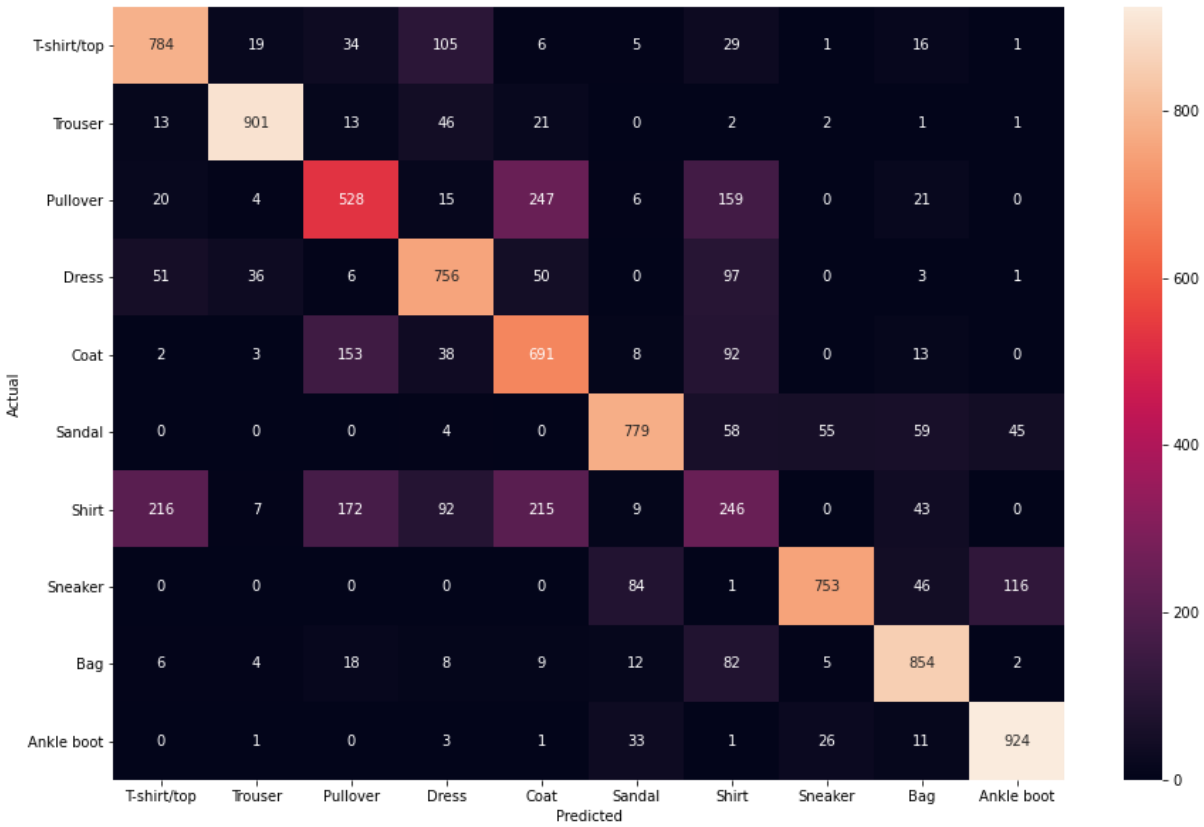

Table 14. Confusion Matrix for Perceptron model with 75% explained variance

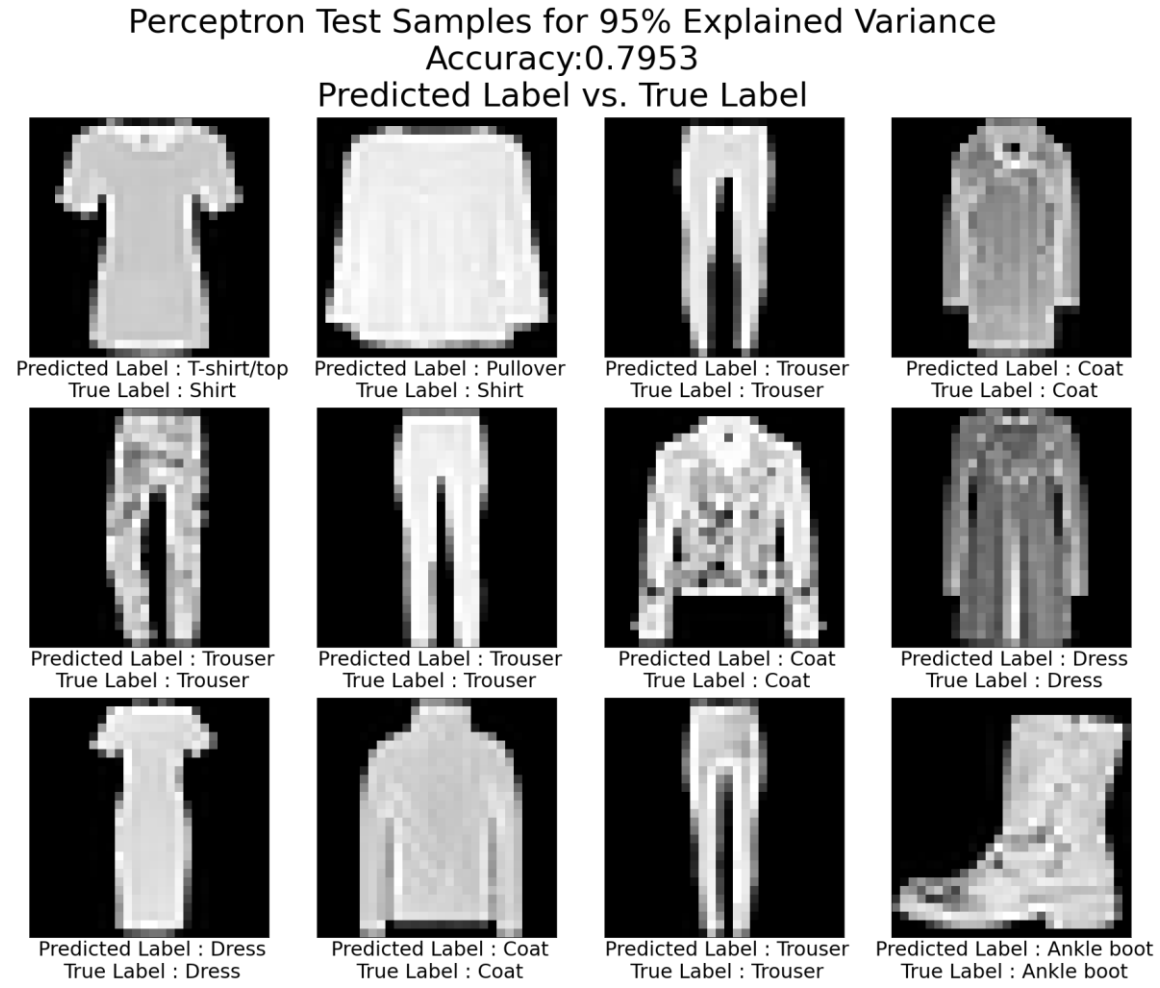

Figure 19. Perceptron sample predictions for 12 images for 95% explained variance

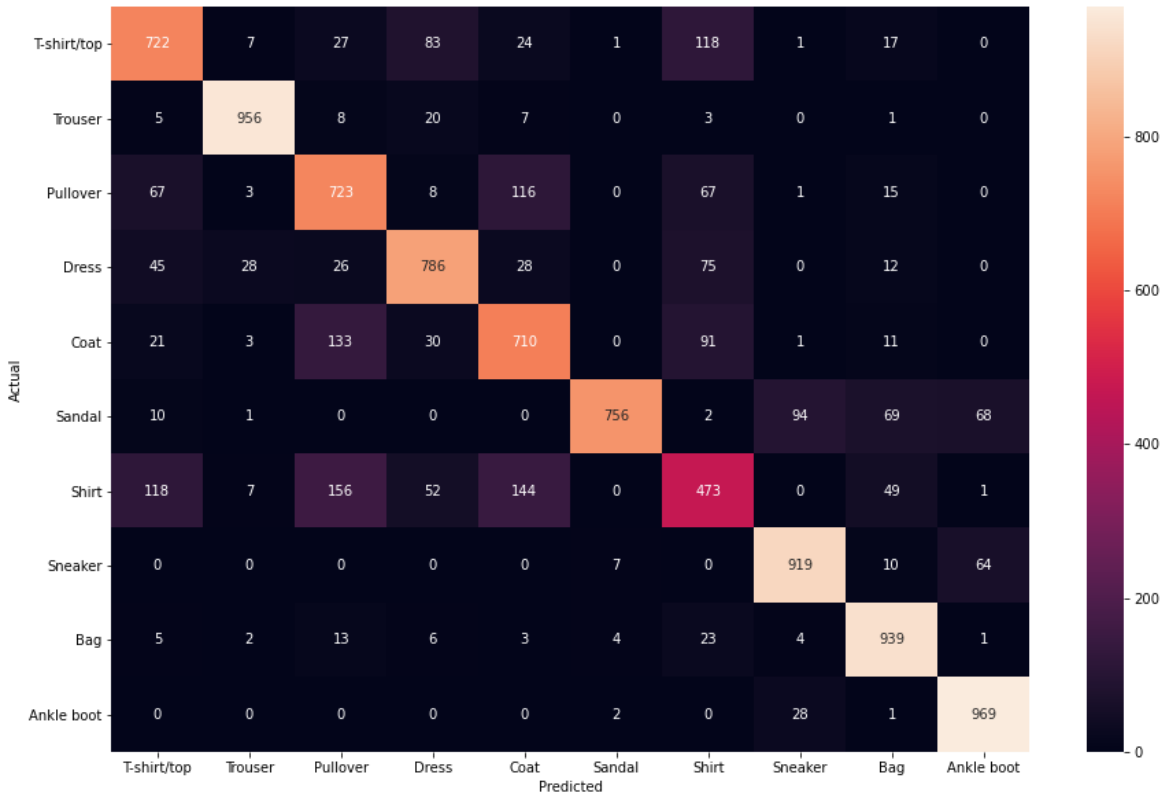

Table 15. Confusion Matrix for Perceptron model with 95% explained variance

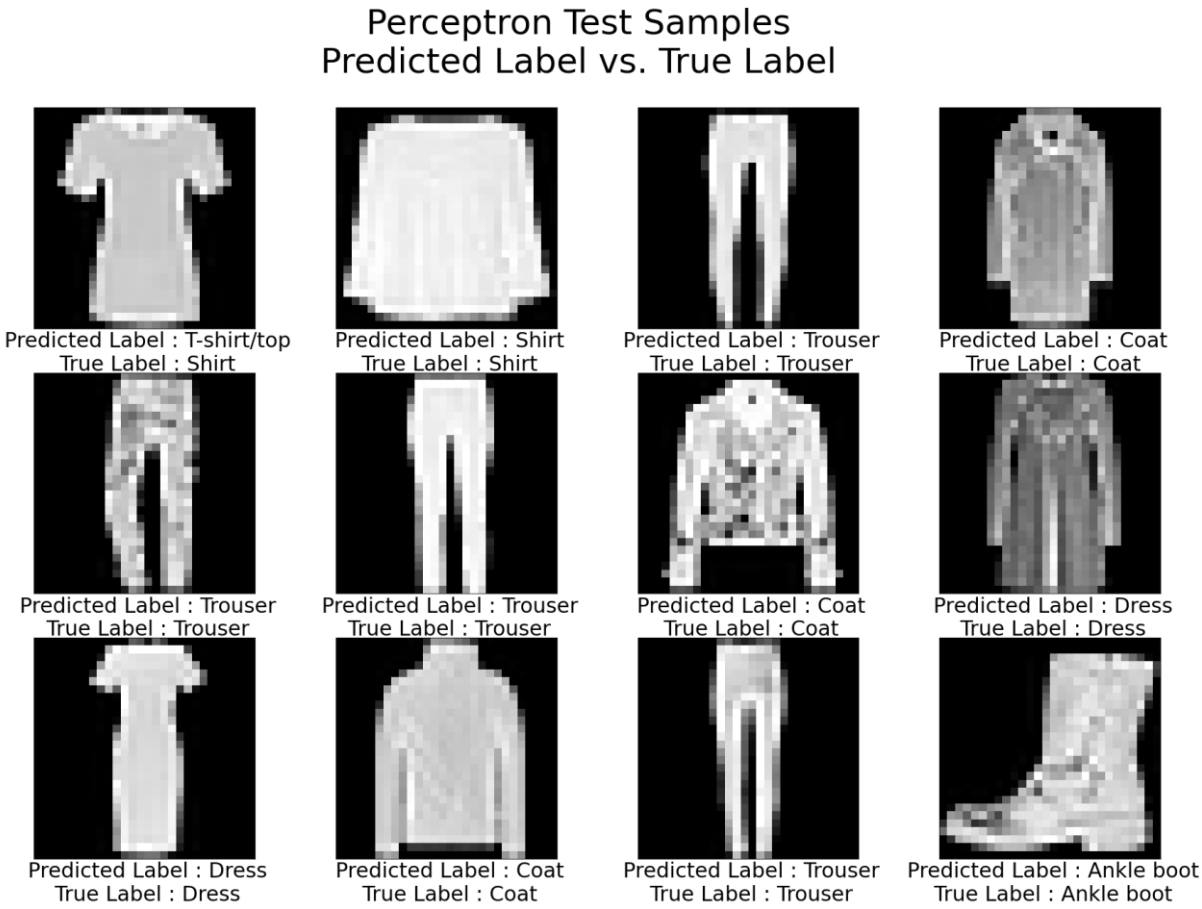

Figure 20. Perceptron sample predictions for 12 images for 100% explained variance

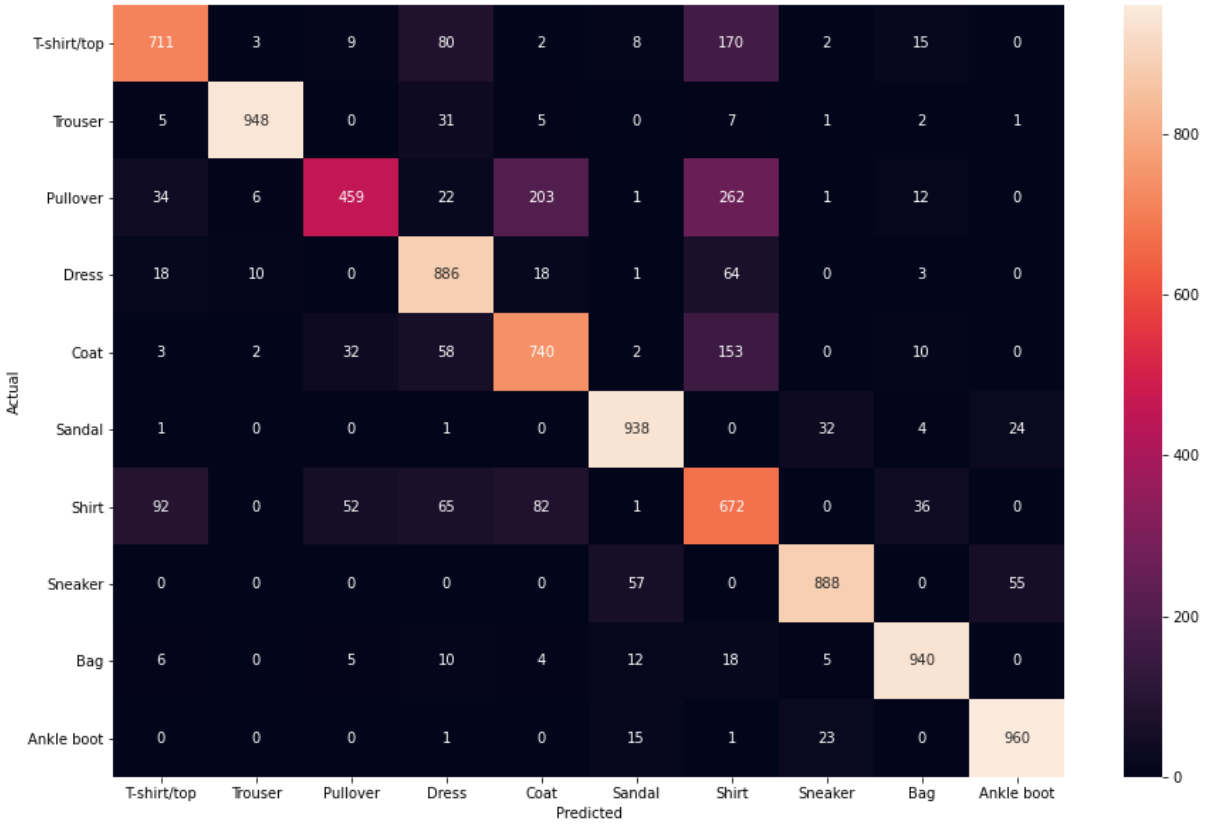

Table 16. Confusion Matrix for Perceptron model with 100% explained variance

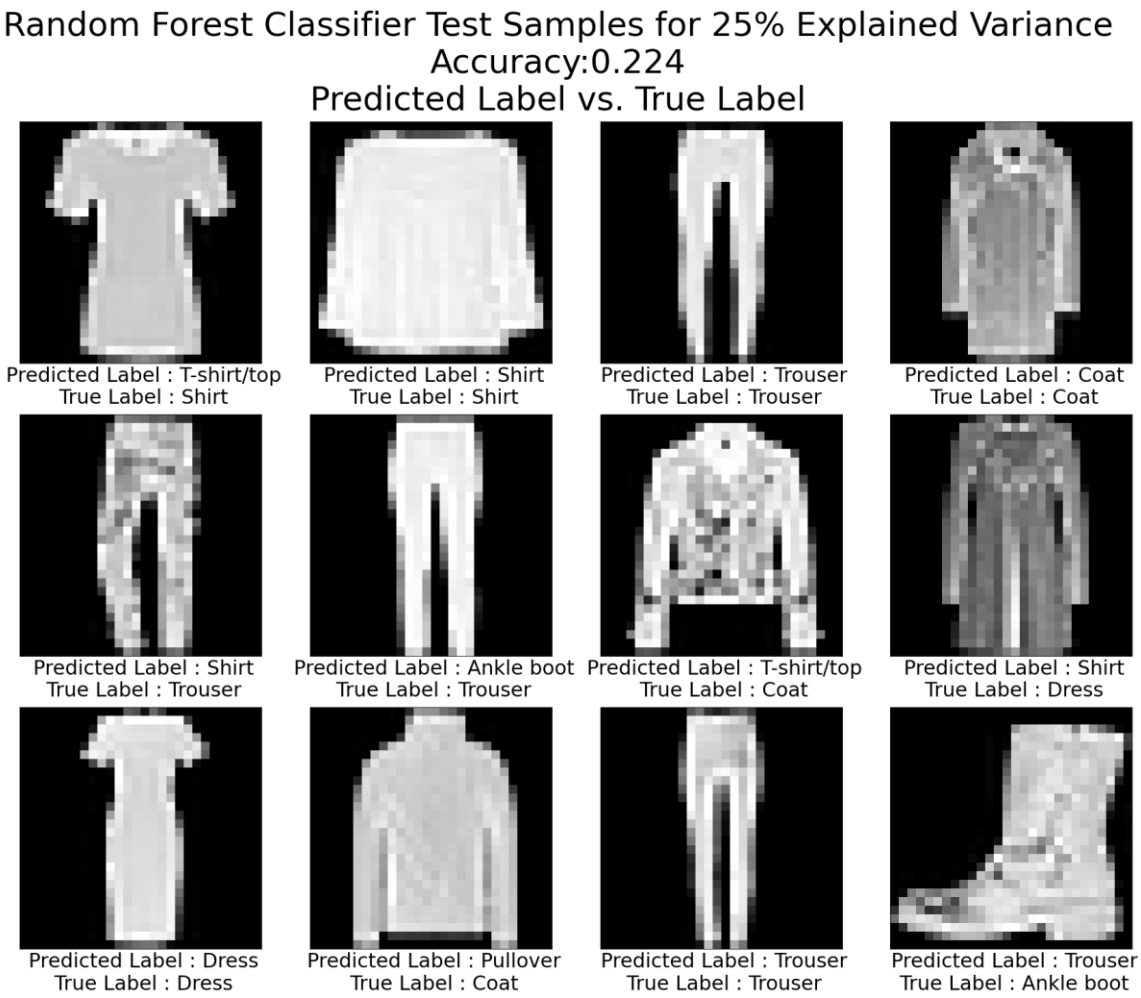

Figure 21. Random Forest sample predictions for 12 images for 25% explained variance

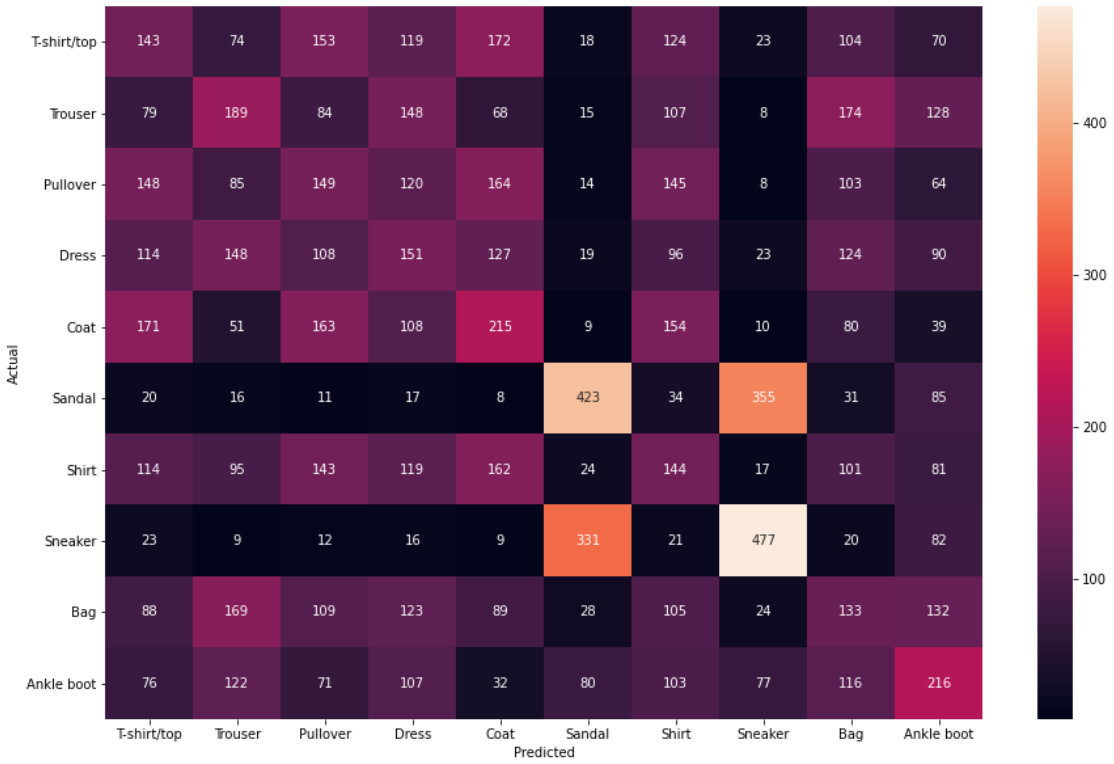

Table 17. Confusion Matrix for Random Forest model with 25% explained variance

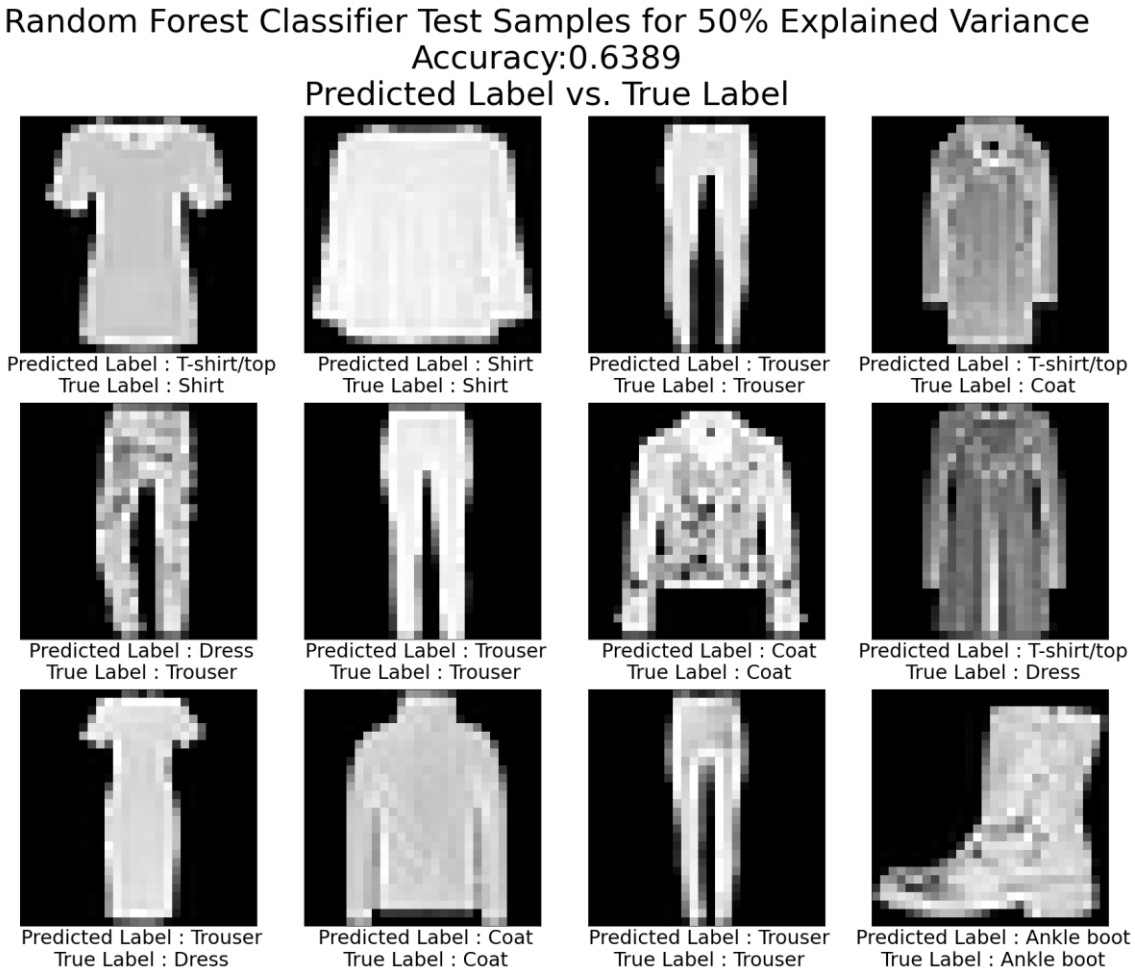

Figure 22. Random Forest sample predictions for 12 images for 50% explained variance

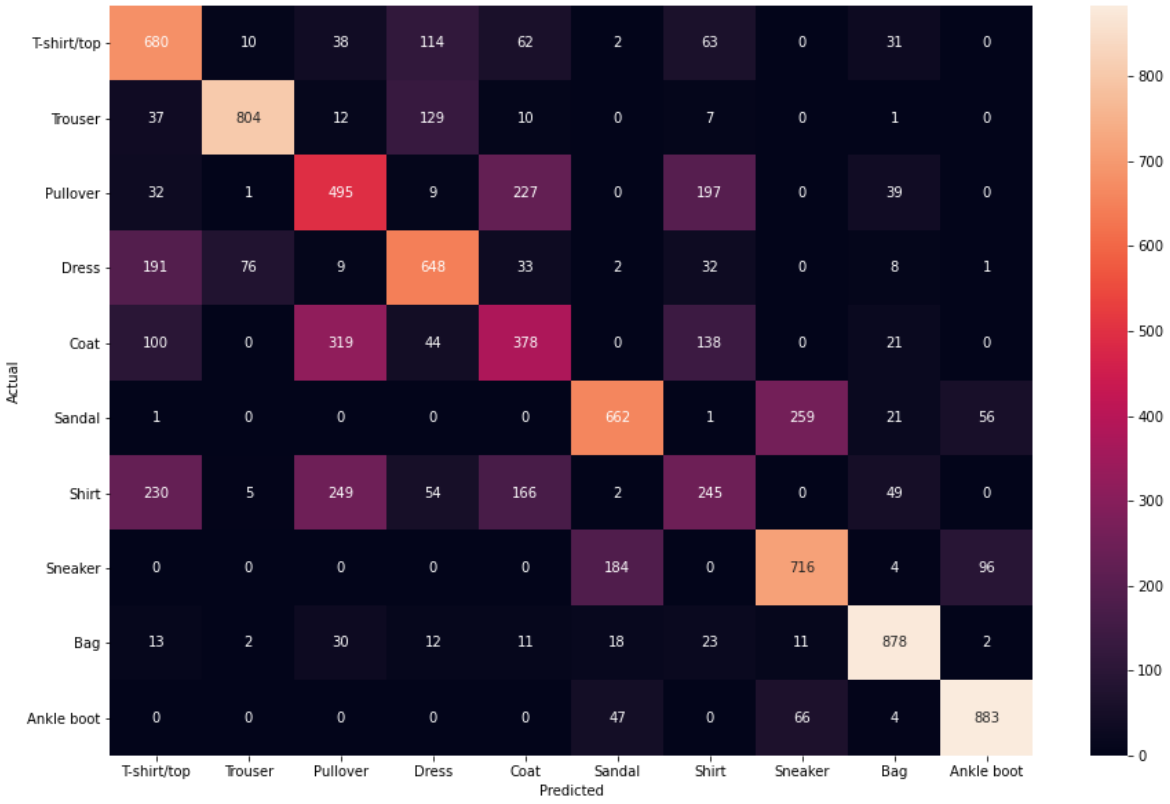

Table 18. Confusion Matrix for Random Forest model with 50% explained variance

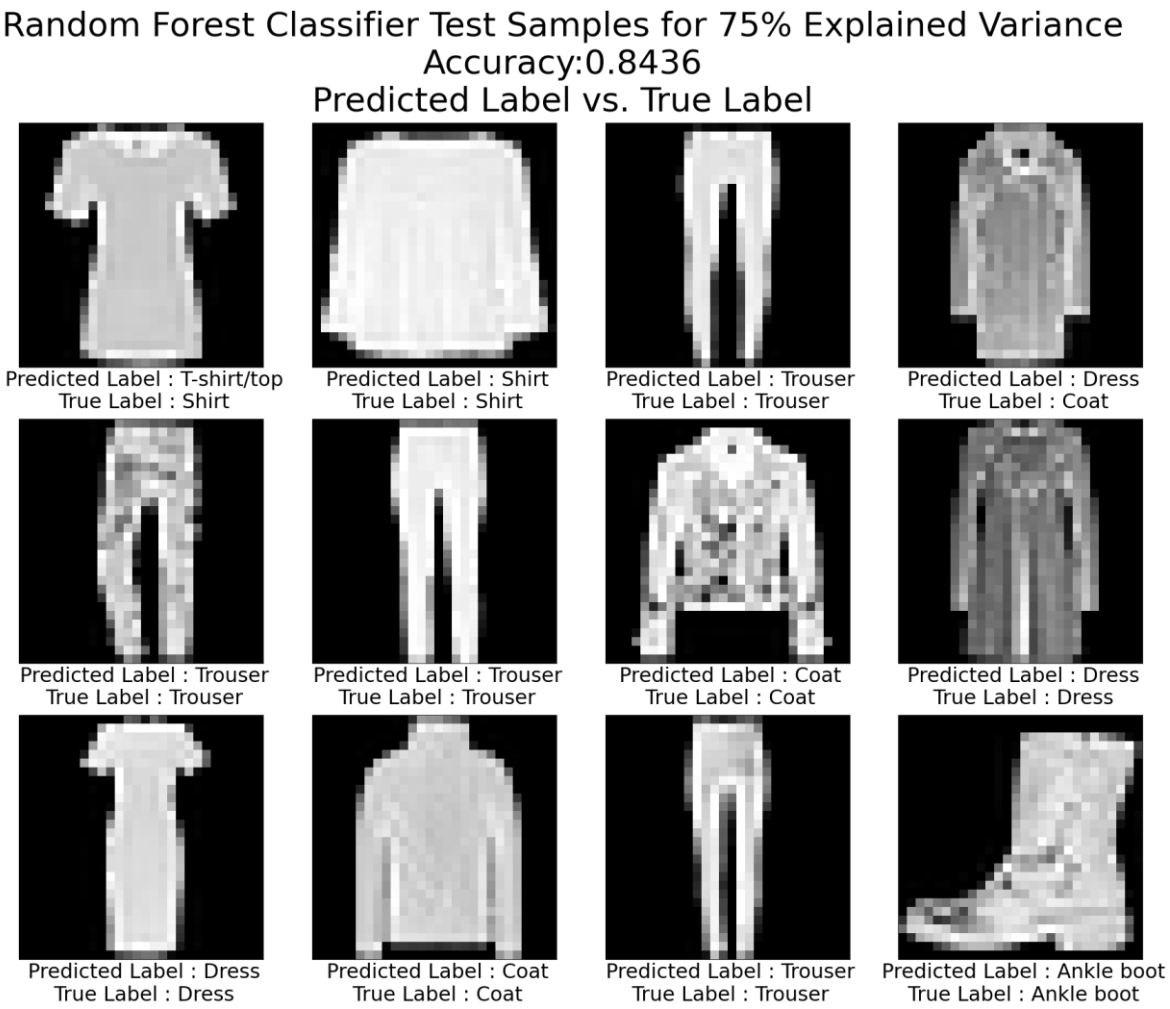

Figure 23. Random Forest sample predictions for 12 images for 75% explained variance

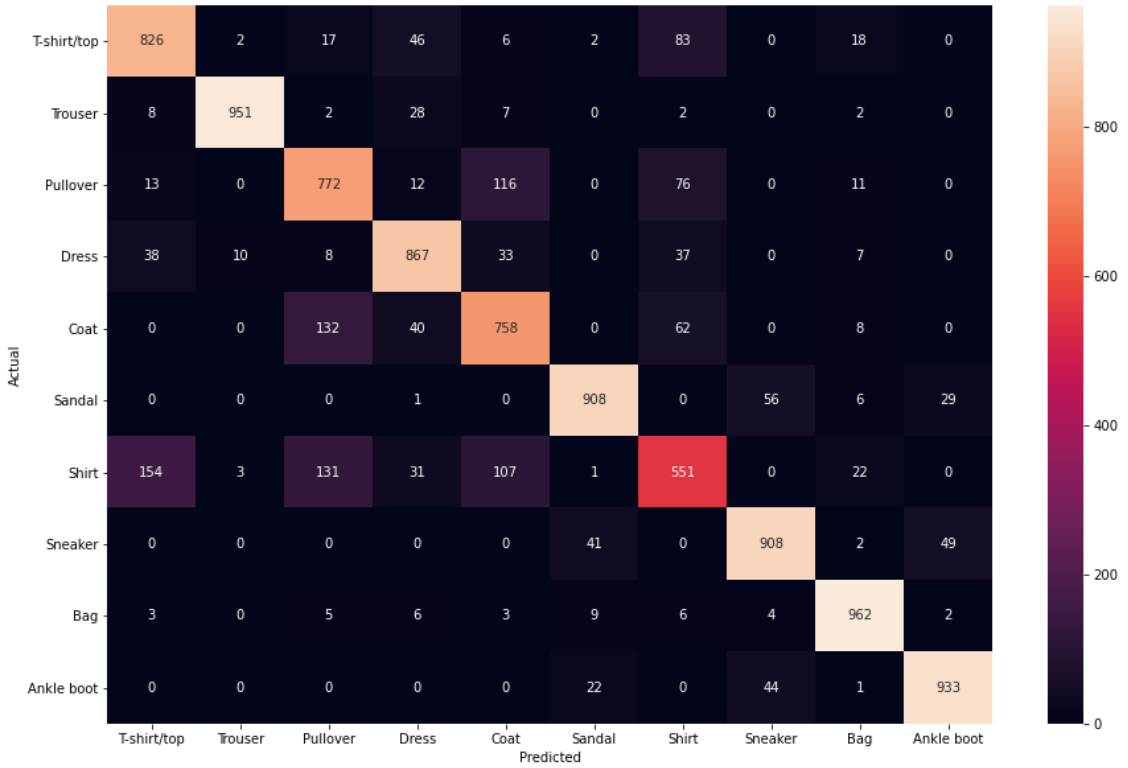

Table 19. Confusion Matrix for Random Forest model with 75% explained variance

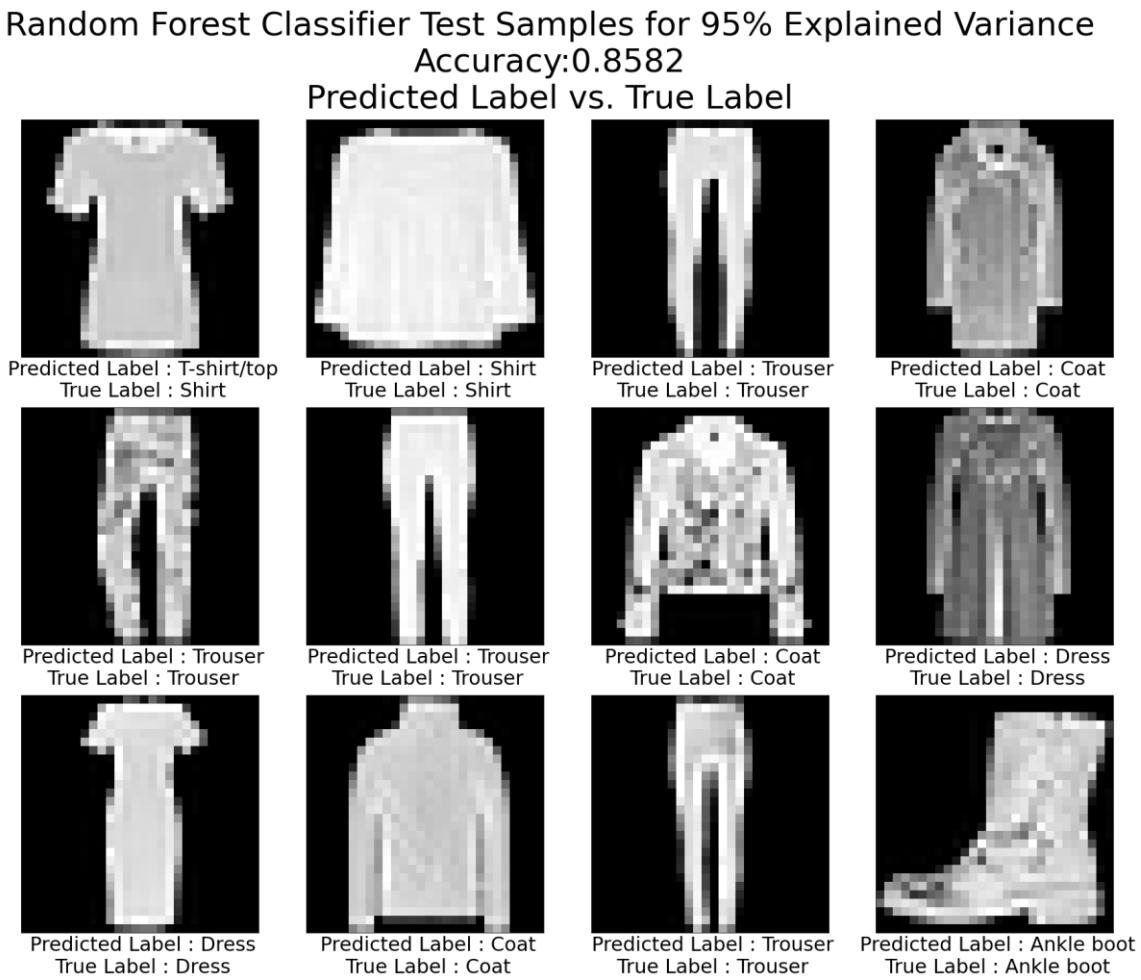

Figure 24. Random Forest sample predictions for 12 images for 95% explained variance

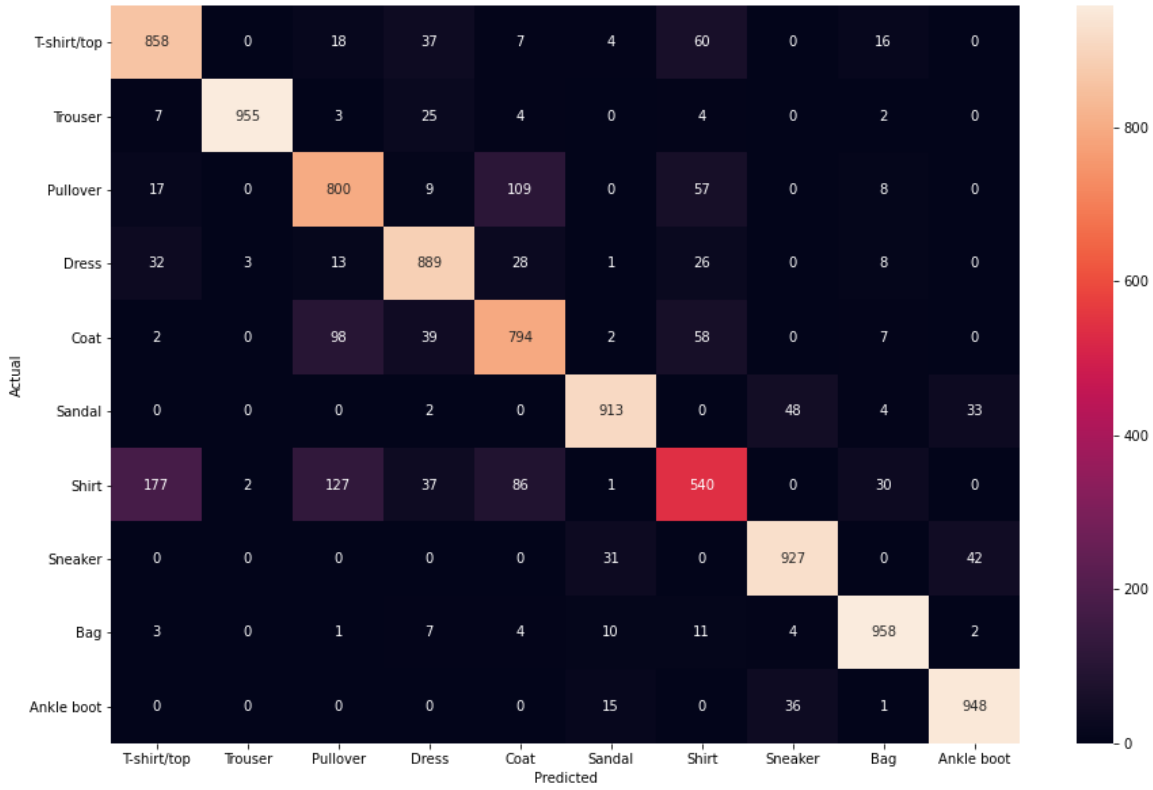

Table 20. Confusion Matrix for Random Forest model with 95% explained variance

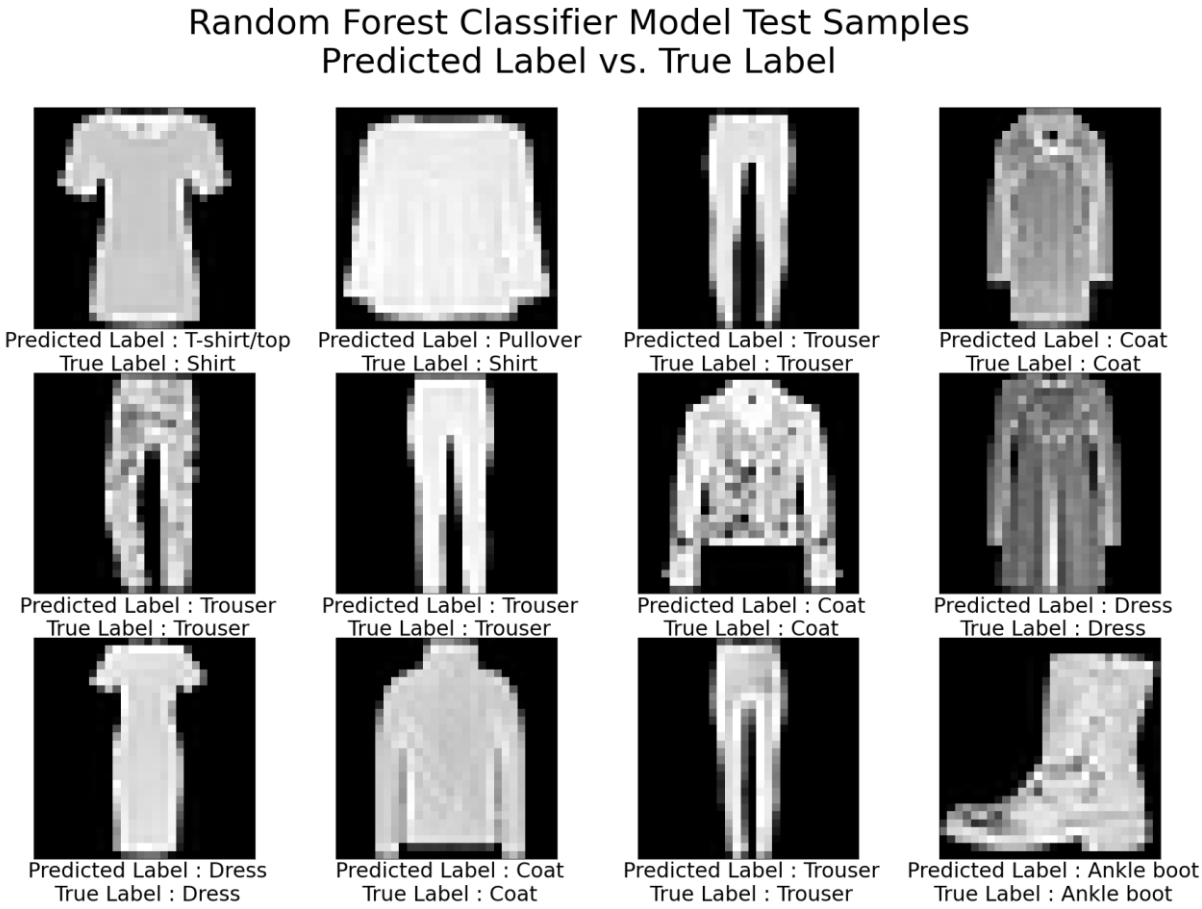

Figure 25. Random Forest sample predictions for 12 images for 100% explained variance

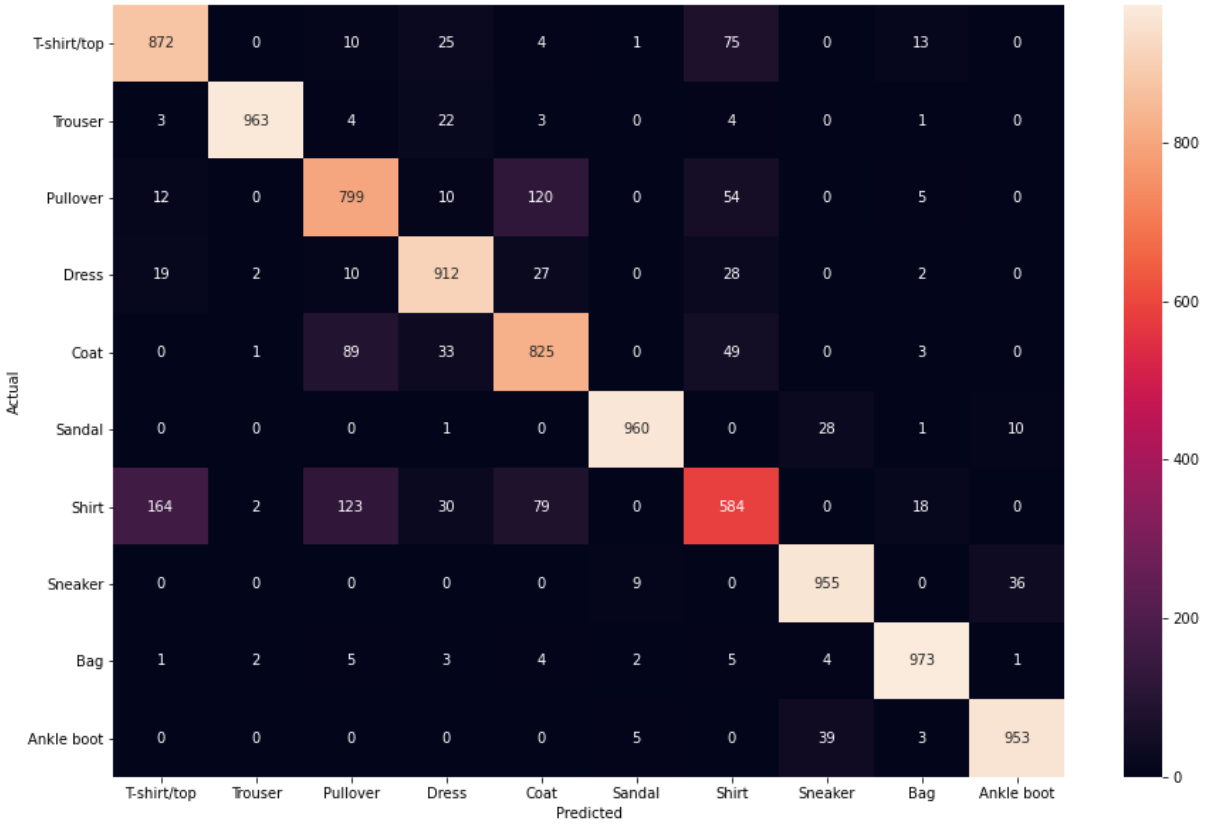

Table 21. Confusion Matrix for Random Forest model with 100% explained variance

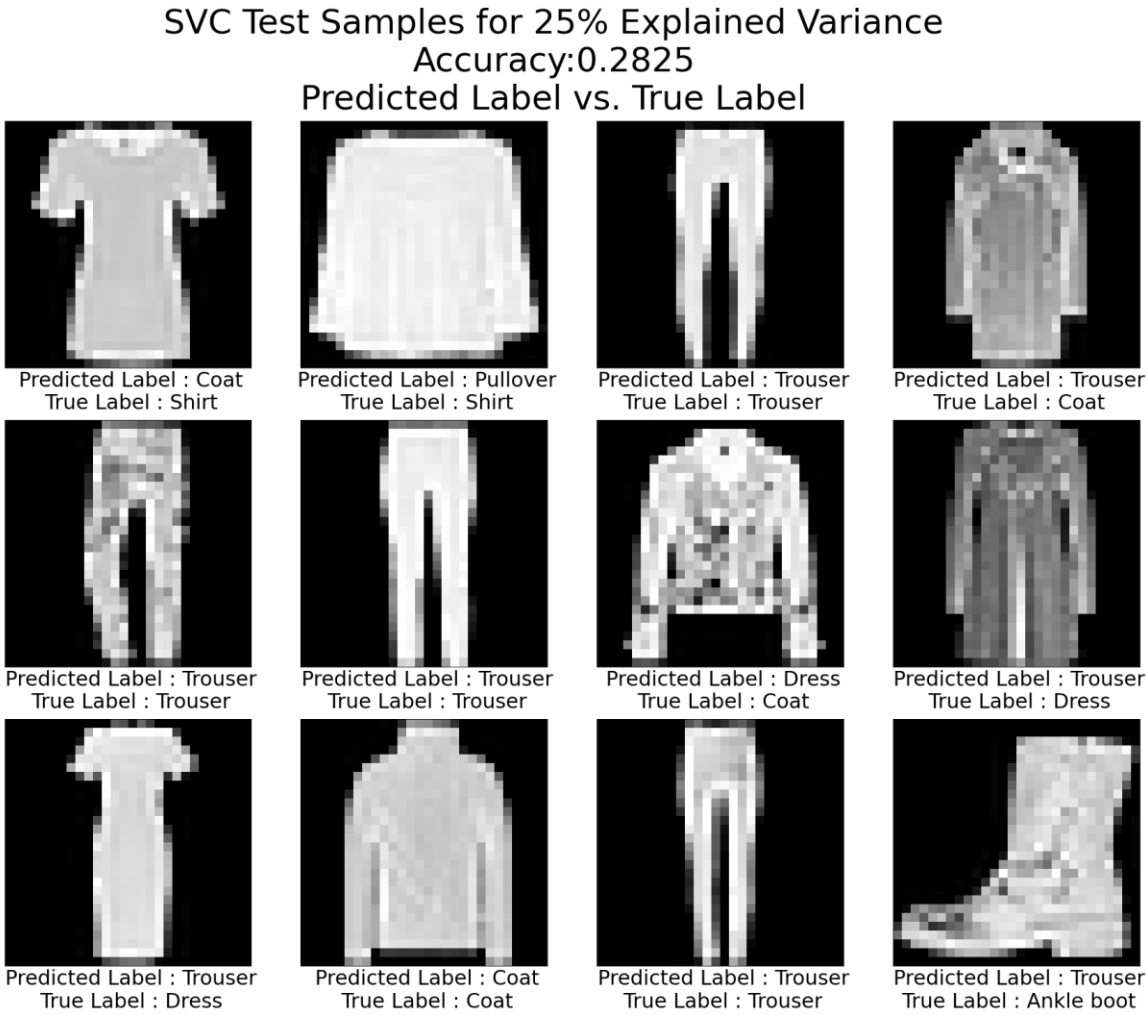

Figure 26. SVC sample predictions for 12 images for 25% explained variance

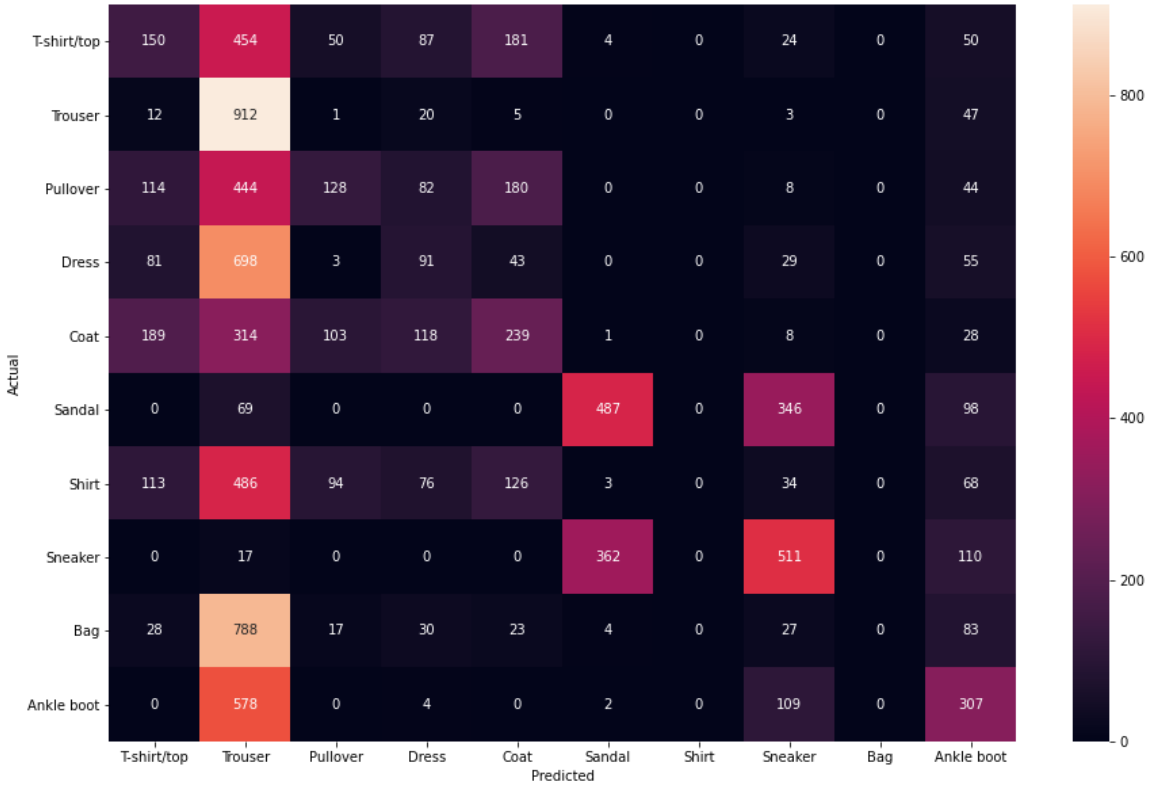

Table 22. Confusion Matrix for SVC model with 25% explained variance

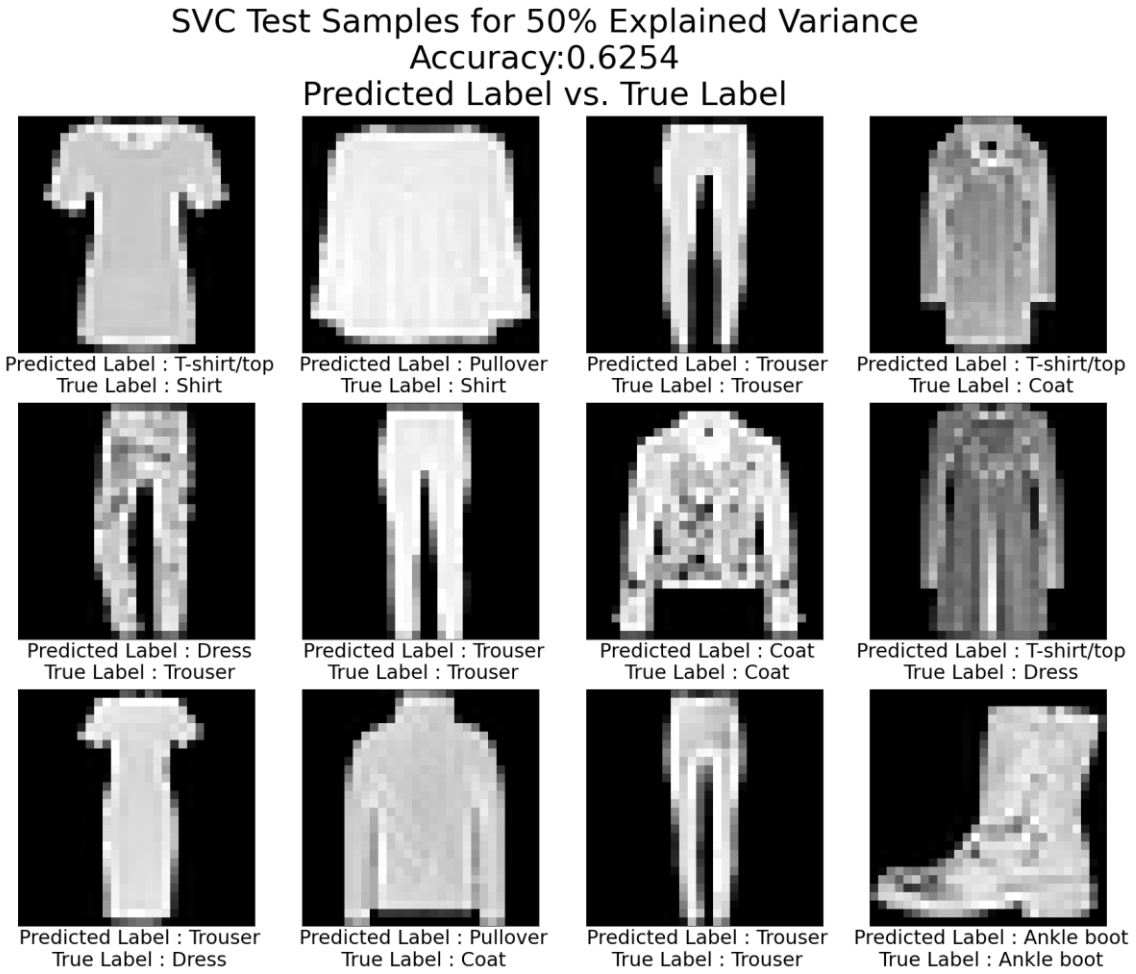

Figure 27. SVC sample predictions for 12 images for 50% explained variance

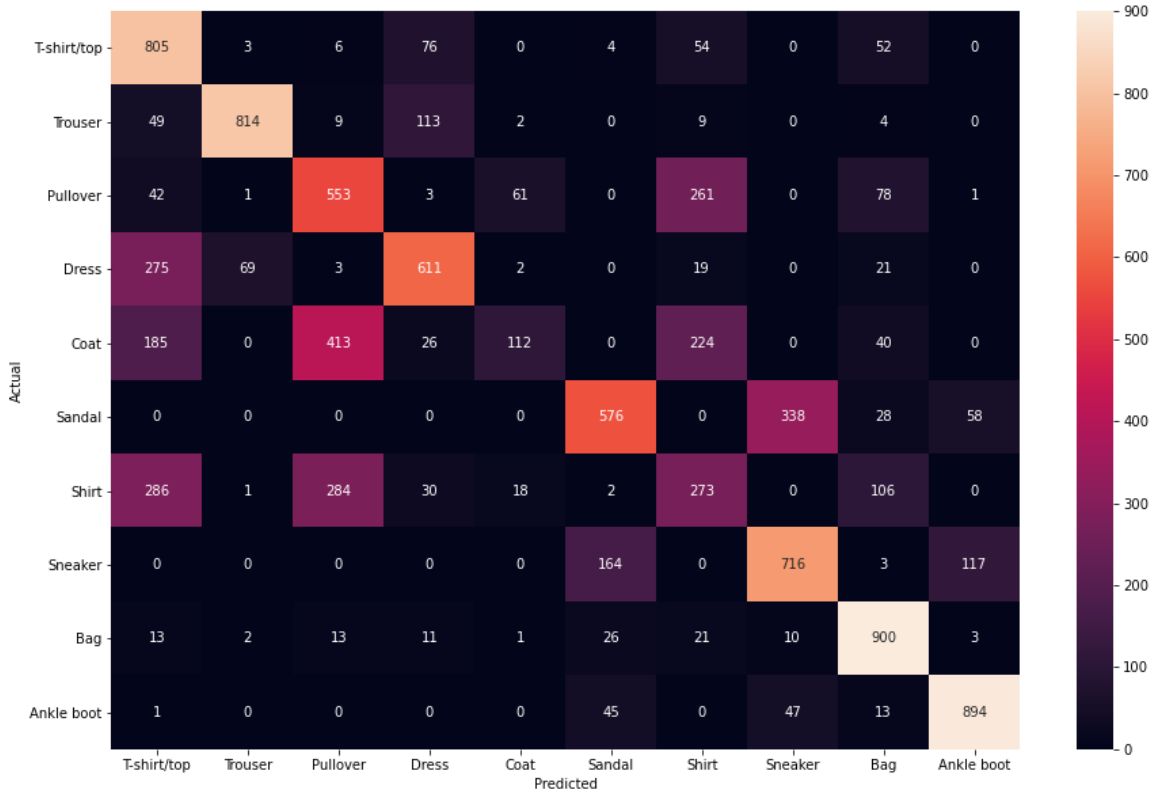

Table 23. Confusion Matrix for SVC model with 50% explained variance

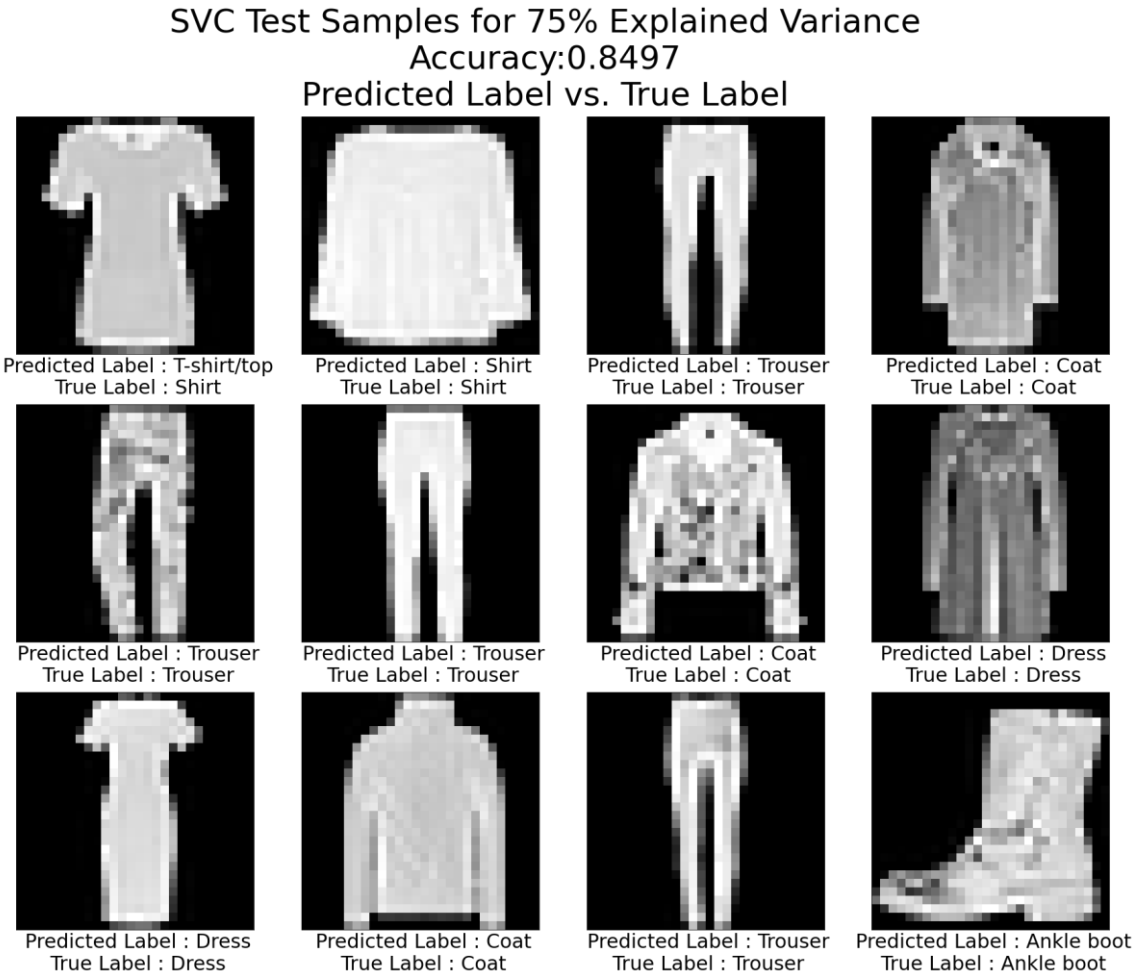

Figure 28. SVC sample predictions for 12 images for 75% explained variance

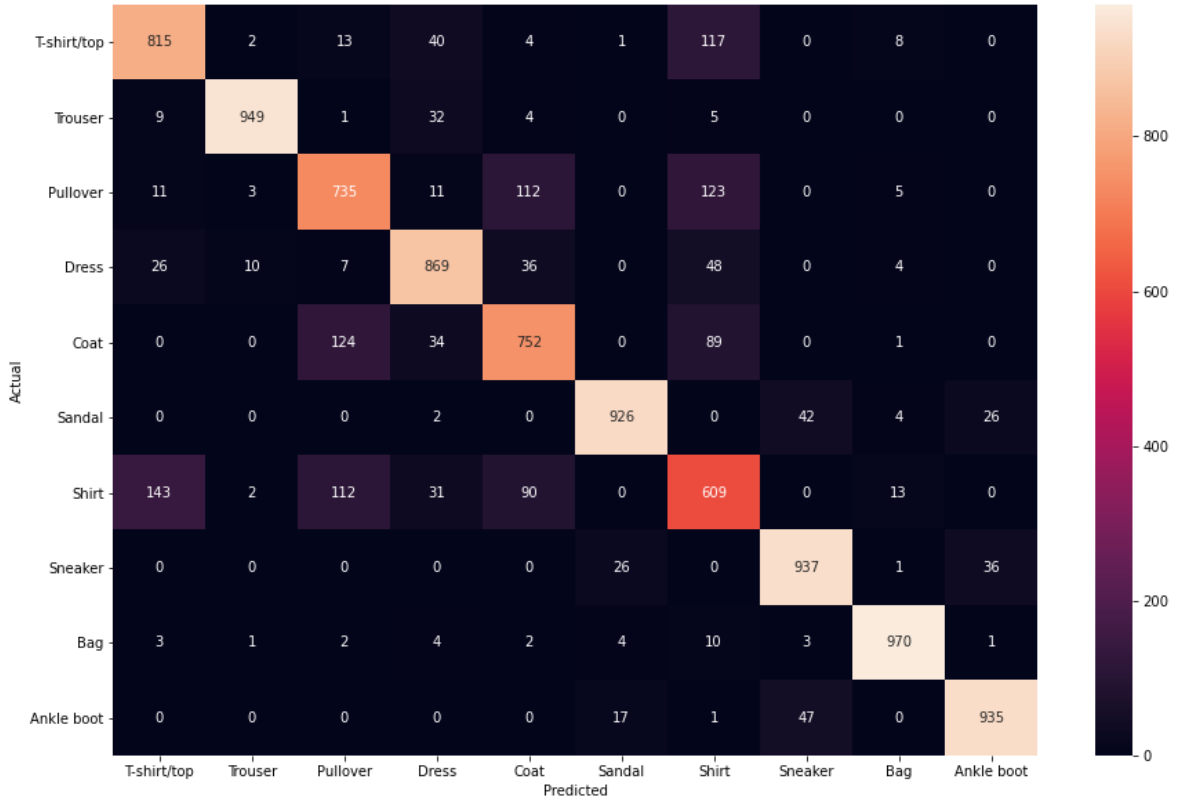

Table 24. Confusion Matrix for SVC model with 75% explained variance

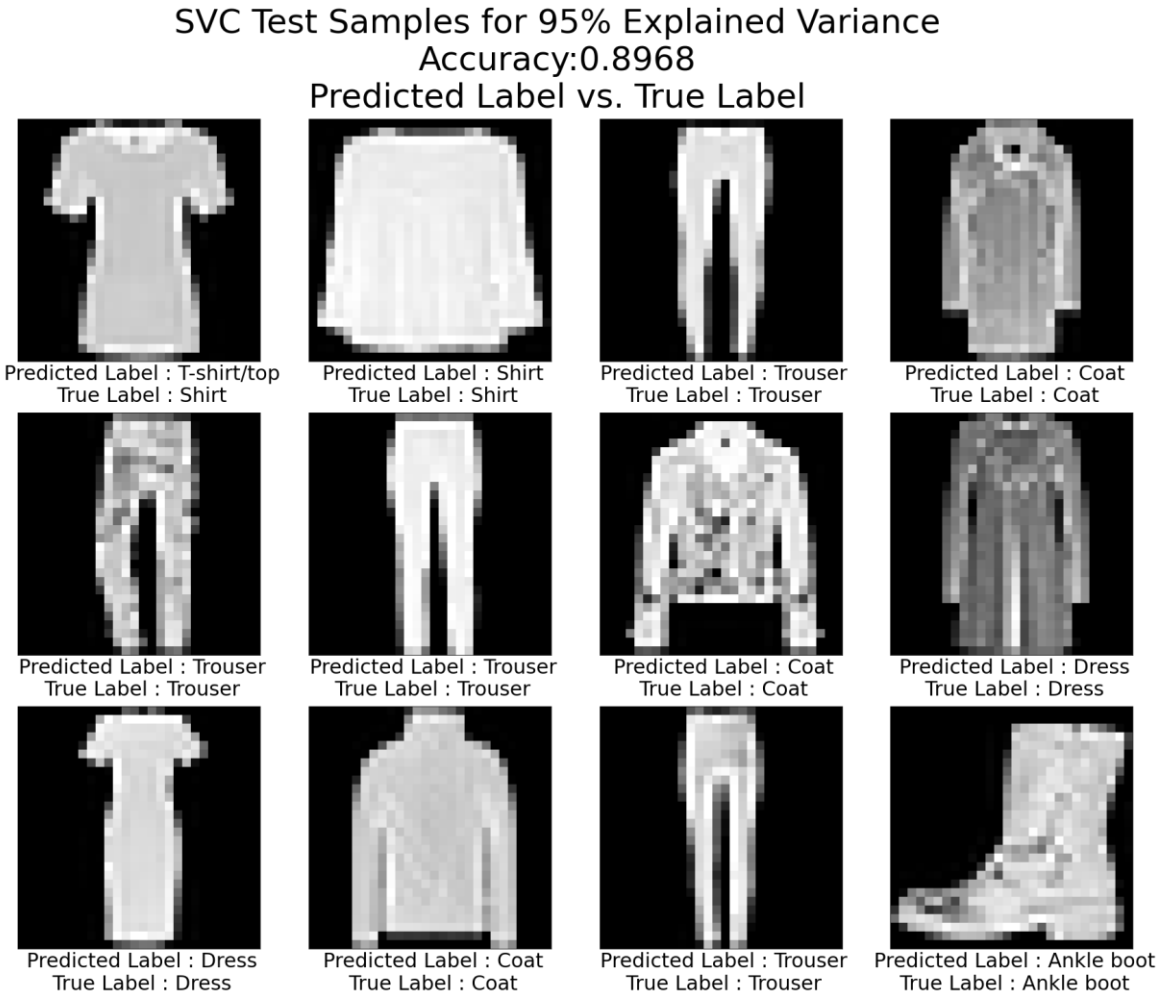

Figure 29. SVC sample predictions for 12 images for 95% explained variance

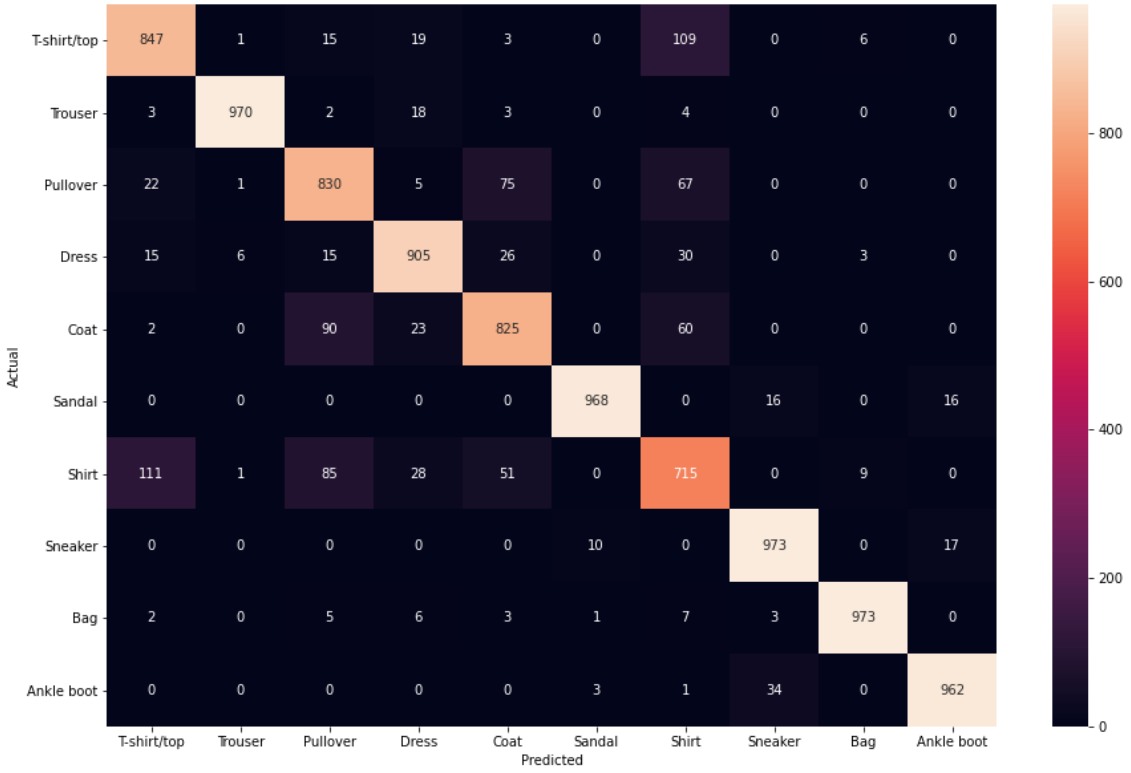

Table 25. Confusion Matrix for SVC model with 95% explained variance

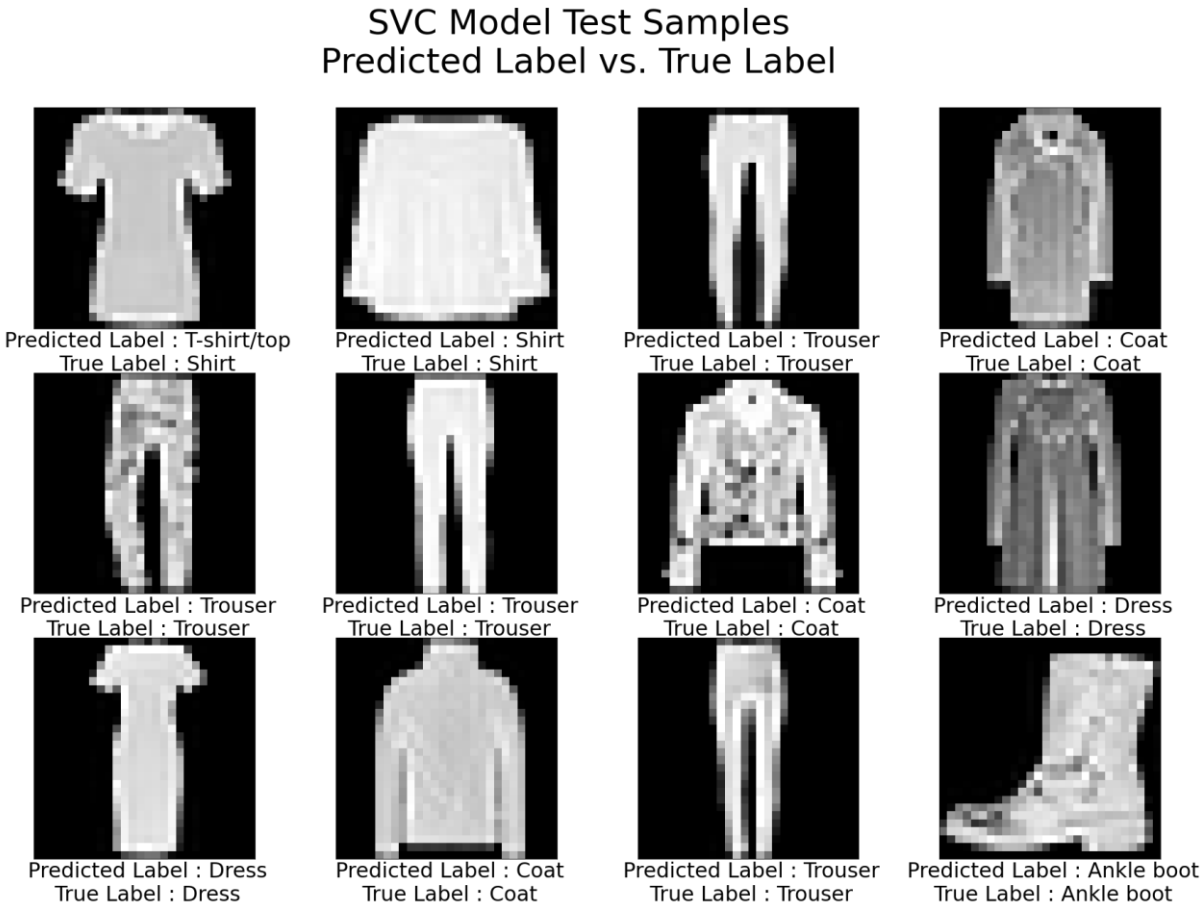

Figure 30. SVC sample predictions for 12 images for 100% explained variance

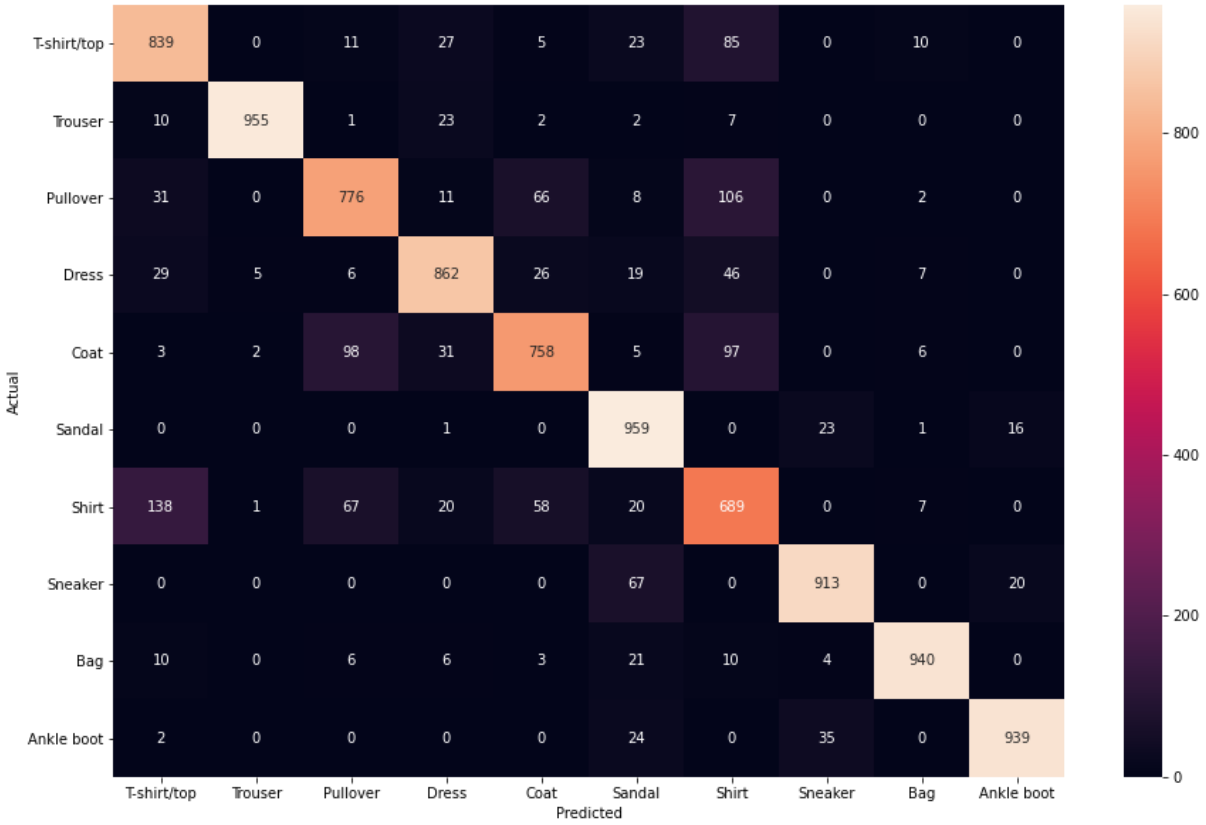

Table 26. Confusion Matrix for SVC model with 100% explained variance

References

[1] Onel Harrison (2018). Machine Learning Basics with the K-Nearest Neighbors Algorithm. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[2] Simplilearn. What is Percetron https://www.simplilearn.com/what-is-perceptron-tutorial#:~:text=The%20Perceptron%20receives%20multiple%20input,the%20class%20of%20a%20sample.

[3] Milecia McGregor (2020). SVM Machine Learning Tutorial https://www.freecodecamp.org/news/svm-machine-learning-tutorial-what-is-the-support-vector-machine-algorithm-explained-with-code-examples/