Q: Aren't the single layer perceptron and the logistic regression model learning the exact same function?

No, they are different loss functions. See here for the mathematical formalizations of the two loss functions (Section 1.5.7. Mathematical Formulation - perceptron vs. log): https://scikit-learn.org/stable/modules/sgd.html. I found this worksheet online (p.3) that may help visualize the differences: https://www.cs.utexas.edu/~gdurrett/courses/sp2020/perc-lr-connections.pdf.

There are many other reasons why you got different results for LR and SGDClassifier in this practical. By default, sklearn LogisticRegression() function uses the 'lbfgs' solver, which is a different optimizer than sgd. SGDClassifier is a generalized linear classifier that will use Stochastic Gradient Descent (sgd) as a solver. Also, you're using L2 penalty in LR vs. none in SGDClassifier. It's important to pay attention to the parameters when setting up a function.

See how it's printed to check which solver you are using:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, l1_ratio=None, max_iter=100,
        multi_class='auto', n_jobs=None, penalty='l2',
        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
        warm_start=False)
```

Q: What is the difference between SGDClassifier and Logistic Regression?

SGD is an optimization method, while Logistic Regression is a model. ML model defines a loss function, and the optimization method minimizes/maximizes it. This can be confusing because it's called "SGDClassifier()" in sklearn, but you can specify a loss function within the SGDClassifier to train an ML model, e.g. "log" would make it a logistic regression model. See the documentation: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

Q: Why would a random classifier give a point along the diagonal line if we are plotting its ROC curve?

A particular random classifier for a particular dataset would be a point in the ROC plot. To obtain the whole ROC curve, we have to vary the probability with which we assign the positive class, from 0 to 1. The ROC curve is a graphical evaluation of the performance of infinitely many classifiers.

Q: What is gamma in the RBF sampler, and why does it make such a difference?

The sklearn documentation explains this very well: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html.

Intuitively, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

The behaviour of the model is very sensitive to the gamma parameter. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.

When gamma is very small, the model is too constrained and cannot capture the complexity or “shape” of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centres of high density of any pair of two classes.

Q: Why does Naive Bayes perform much worse than the other models?

Naïve Bayes (NB) assumes all the features to be conditionally independent, so if some of the features are in fact dependent on each other (especially in case of a large feature space), the prediction might be poor.
Logistic regression splits feature space linearly and typically works reasonably well even when some of the variables are correlated.

Generally, when the training data size is small relative to the number of features, the information/data on prior probabilities used in NB would help in improving the results. In larger data sets, logistic regression with regularization can help reduce overfitting and often results in a more generalised model.

Q: Why does the kernel trick improve the accuracy for some models and reduce the accuracy for others?

Kernel trick draws a non-linear decision boundary by mapping the data into higher dimension. That works well if the true decision boundary is non-linear, but it risks over-fitting. Given logistic regression performs so well on this data set, it seems the linear boundary fits the data well. Adding the kernel trick is over-fitting to the data and not generalizing to the test data.