Project Name: Telco Churn Classification Project
Project Type: Required solo project

Project Goals:
Find drivers for customer churn.
Construct a ML classification model that accurately predicts customer churn.
Include conclusions
Create modules that make your process repeatable - readme.md, data_dictionary.md, acquire.py, prepare.py, .csv file with all predictions and actual values by customer_id, documented .ipnb with 5 min walk through
Document your process well enough to be presented or read like a report using Jupyter Notebook.

Stage	Tools	Brief Description of Process	Challenge Resolution
Plan	• Visual Studio	• In Visual Studio create a new readme.md file to outline project plan • Import key elements and deliverables from curriculum requirements • Add storytelling elements	• Noted that we were given clearer expectations than usual for this assignment

Acquire	• Visual Studio
	• MySQL
	• .py script
	• SQL query
	• Query function
		• In Visual Studio create a new acquire.py file to obtain data from Codeup Telco Churn database
		• Used scripts I had previously created for other database queries and adapted for this file
		• Note for documentation that user will need access to Codeup databases in order to replicate
		• Use MySQL to write and verify SQL query
			• No unusual challenges in this section
			• Python libraries needed:
			  ○ Pandas
			  ○ Seaborn
			  ○ Matplotlib
			  ○ Numpy
			  ○ Scipy
			  ○ Sklearn

Prepare	• Visual Studio • .py script • Jupyter Notebook • Matplotlib	• In Visual Studio create a new prepare.py file • Started from scripts I had previously created • Assess overall data and determine where columns need to be encoded • Plot individual variables • Asses for null and duplicate values • Convert total_charges from string to float and drop resulting null values • Add years_tenure feature • Add extra_services feature • Changed text response “No Phone Service” and “No Internet Service” to “No” for first iteration • Drop duplicate columns and rename remaining columns • Identify target variable = churn • Split into train, validate, and test datasets for exploration stage	• Drop columns function not working, unable to get this working for this project • Added remove duplicates for future use • Converting to float using .astype not working for total_charges • Googled alternative .to_numeric and added flag errors='coerce' • Did not remove tenure less than 1 month for this iteration • Initially decided to drop Customer_id, but put back in to assist in create predictions.csv
Explore	• Jupyter Notebook • Seaborn • Scipy.stats • Matplotlib	• Recreated prepare visualizations adding churn • This identified Fiber Optic as group for further analysis • Visualize extra_services with monthly charges and churn • This identified tipping point at 4 and under vs 5 and over count of additional services as subgroup for investigation • Set up Hypothesis tests for: do those who pay more, churn more? And: do those who have been with us less time pay more? • Confirmed average monthly payment is higher for those who churn	• Hardest challenge was finding starting point for investigation • Challenging to convert initial hypothesis question into statistically testable hypothesis • Challenging to translate what I want to see into code needed to produce charts and/or metrics

		• Positive linear relationship exists between monthly charges and tenure, but the relationship is weak based on r=.22	
Model	• Jupyter Notebook • Sklearn	• 1st machine learning project so decided to make 1 model of each classification type using all features and primarily default hyperparameters • Established Baseline accuracy • Decision Tree set max depth as half of the number of features to reduce potential overfit • Random Forest set max depth at ¾ of features and increase min leaf to 5, left n_estimators at 100 • Logistic Regression used default hyperparameters • K Nearest Neighbors set K=¼ of features	• No unusual challenges in this section
Evaluate	• Jupyter Notebook • Sklearn	• Focused solely on accuracy as measure of performance in this iteration • Found Random Forest and Decision Tree had best performance on training dataset • Ran both models on validate dataset and found Decision Tree accuracy performance declined indicating it was overfit on training data • Ran only Random Forest on test dataset • Found consistent accuracy of 87%-88% for this model on all 3 datasets • That is a significant improvement over the baseline accuracy: 73%	• No unusual challenges in this section

Model Explanation	**How does your algorithm work?** (For example, if your best ML model used a Random Forest Classifier, how does that algorithm work?)	● Random forest is a type of ensemble ML algorithm called Bootstrap Aggregation or bagging. ● You take lots of samples of your data, calculate the mean, then average all of your mean values to give you a better estimation of the true mean value ● Multiple samples of your training data are taken and models are constructed for each sample set ● When you need to make a prediction for new data, each model makes a prediction and the predictions are averaged to give a better estimate of the true output value. ● Random forest is a tweak on this approach where decision trees are created so that rather than selecting optimal split points, suboptimal splits are made by introducing randomness.	● I think of Random Forest as taking many Decision Trees and then combining their predictions results in a better estimate of the true underlying output value.
Delivery	● Jupyter Notebook ● .csv file	● Used combination of markdown cells in Jupyter Notebook and commented out text within code cells to document process and explain code and charts ● Added agenda, Executive Summary, and Conclusions for walk through	● Had to figure out how to get predictions, actuals, and customer_id into single pandas dataframe then write to .csv file ● Markdown in Jupyter Notebook doesn't always print correctly once uploaded to github ● 5 minute presentation time was particularly challenging to stay within