**Machine Learning Project Documentation**

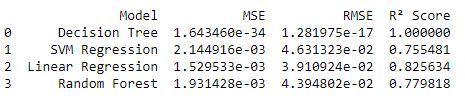
**Model Refinement**

**1. Overview**

There are opportunities to further enhance the machine learning model. The objective of model refinement is to improve the accuracy, reliability, and generalization capabilities of the machine-learning model. For instance, employing techniques such as feature selection, hyperparameter optimization, feature selection, cross-validation or ensemble methods may have the potential to elevate the model's performance.

**2. Model Evaluation**

We conducted a model selection process to evaluate and compare the performance of various machine learning algorithms, including Decision Tree, SVM Regression, Linear Regression, and Random Forest. Based on the evaluation metrics, we determined that the Linear Regression model exhibited the most desirable characteristics. Specifically, it had the smallest Mean Squared Error value, which was not excessively close to zero, indicating good fit. Additionally, the model demonstrated strong performance on other key metrics, such as Root Mean Squared Error and R-squared Score, suggesting its ability to avoid overfitting. Notice that this process was carried out using training data.



Summarize the initial model evaluation results and highlight areas for improvement.

Reference key metrics and visualizations from the model exploration phase.

**3. Refinement Techniques**

Describe the techniques used for refining the model. This may include adjusting

hyperparameters, trying different algorithms, or incorporating ensemble methods.

To avoid overfitting, we used regularization techniques, such as the Ridge function from scikit-learn, and tuned the hyperparameters using GridSearchCV to determine the optimal alpha value. Prior to this, we evaluated several algorithms mentioned earlier and selected the best-performing model based on the provided metrics. We then explored various techniques to further enhance the model's performance.

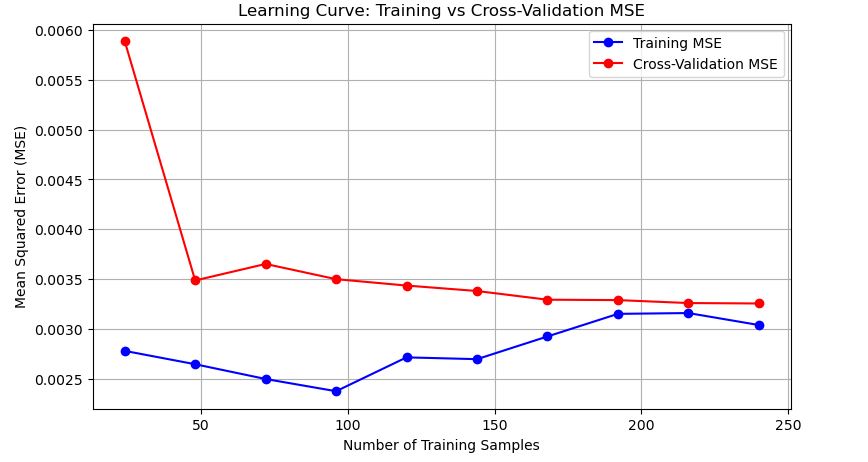
**4. Hyperparameter Tuning**

Detail any additional hyperparameter tuning performed during the refinement phase. Include

insights gained and their impact on the model's performance.

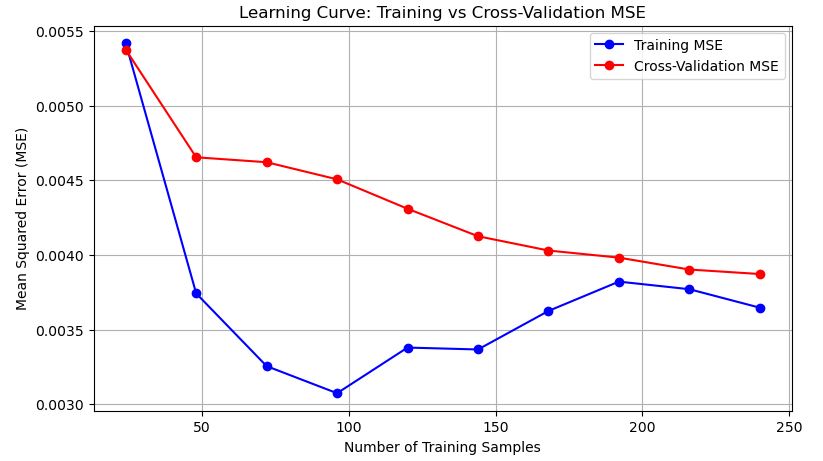
We use the learning curve from scikit-learn to assess the performance of our model before and after applying ridge regularization.

Before ridge regularization:



The initial high cross-validation error and low training error indicate overfitting when the dataset is small. As the number of samples increases, both errors decrease and converge, indicating improved generalization. In this case, the model seems to reach a good balance, with both errors stabilizing at relatively low values.

After ridge regularization:



Early overfitting is indicated by the initial gap between training and validation errors, especially with small sample sizes. As more data is added, the gap decreases, and the model generalizes better, as indicated by the cross-validation MSE steadily or consistently dropping. The model is improving its performance with more training data, indicating an optimal balance between training and validation errors. Finally, we considered the first one to deploy our model.

**5. Cross-Validation**

I used both 5-fold and 10-fold cross-validation to assess model performance in the function GridSearchCV. The results were very similar, indicating that the model is generalizing well, and increasing the number of folds did not lead to any significant performance improvement. It suggests the model is generalizing well.

**6. Feature Selection**

To improve the model's performance, I applied the PolynomialFeatures function from the scikit-learn library to create polynomial features. This allowed me to capture nonlinear relationships between the input variables and the target variable, which could potentially enhance the model's predictive accuracy.

**Test Submission:**

**1. Overview**

After comparing the results for the training, cross-validation, and test sets with the metrics provided, we judge that the model is ready for deployment. We used the scikit-learn pipeline to prepare the model for deployment. Additionally, we utilized the pickle module to serialize the model object into a binary representation, which can be stored in memory before using the FastAPI framework.

**2. Data Preparation for Testing**

After splitting the data using the train\_test\_split function, we performed the following steps:

* Polynomial Feature Creation: If polynomial features were used during training, we ensured to apply the same transformation to the testing data.
* Standardization: We scaled the features to the same range or distribution as used during training.  
  Then, we made predictions on the test set and evaluated the model's performance with metrics.

**3. Model Application**

* We loaded the trained model and saved it using Pickle, which was then loaded to make predictions on the test dataset.
* Next, we prepared the test data and fed it into the model to generate predictions.

**4. Test Metrics**

This shows the model fits the training data quite well. An **R² score** of 0.837 indicates that the model explains around 84% of the variance in the training data.

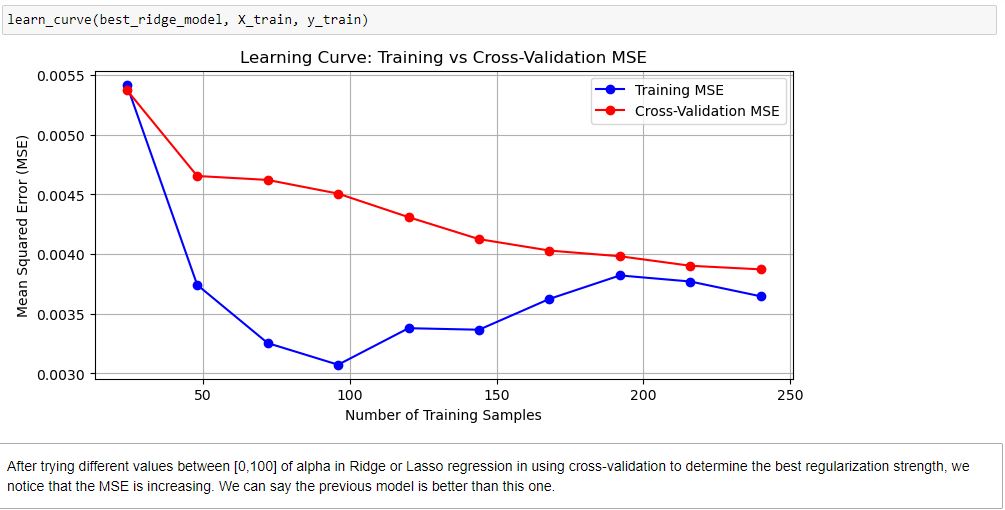
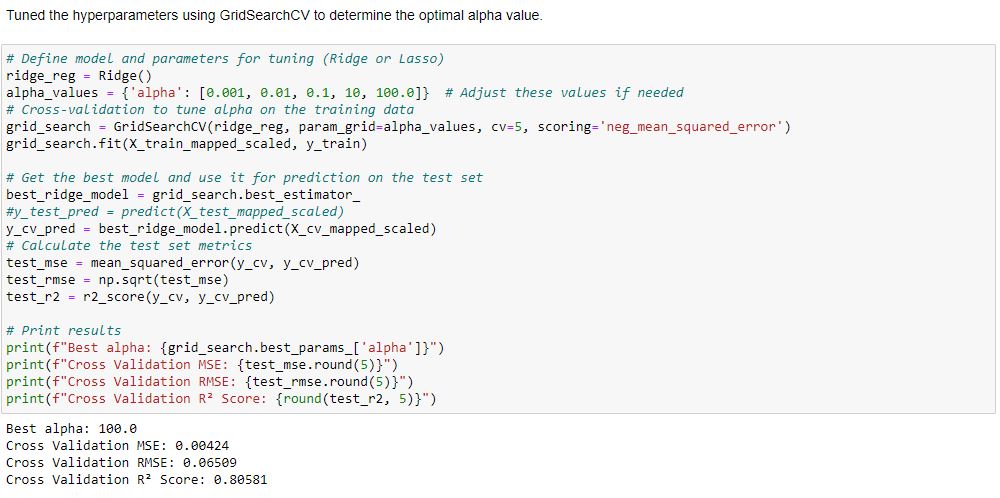
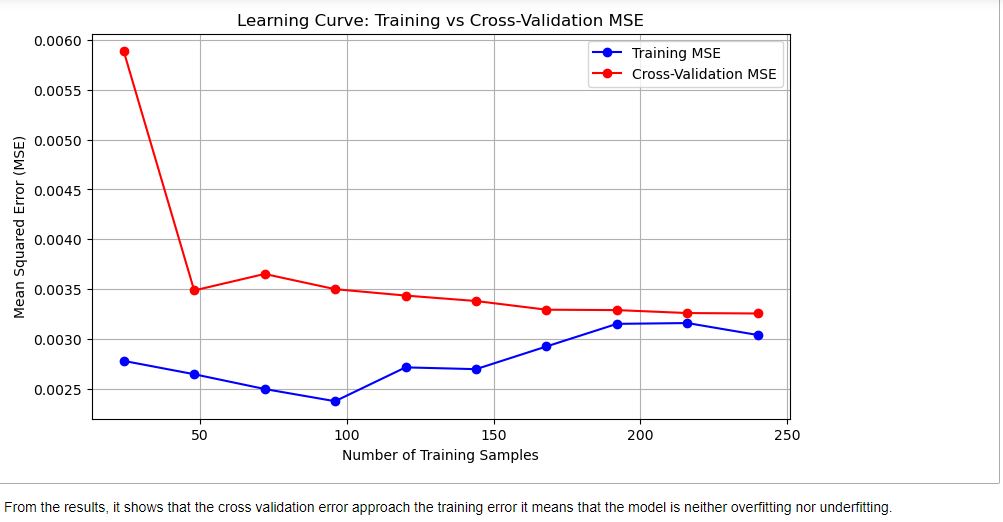
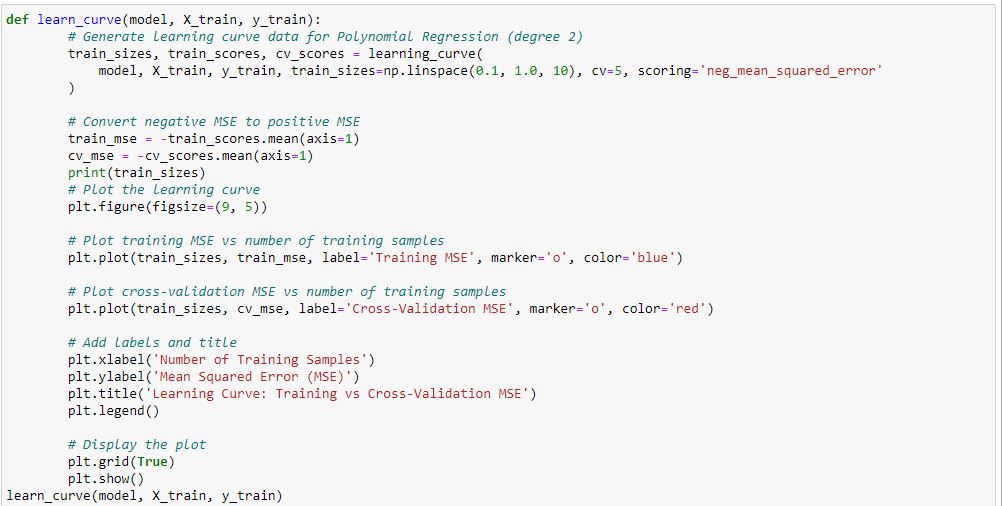
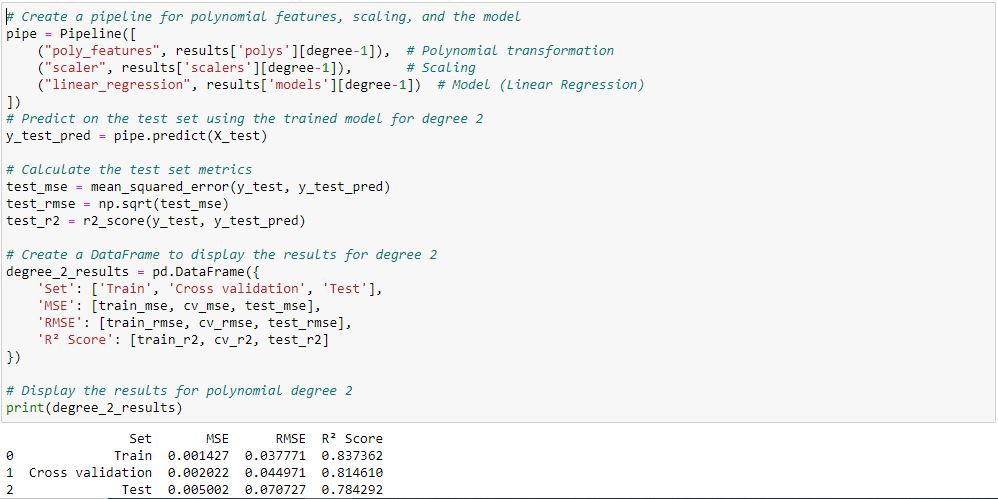
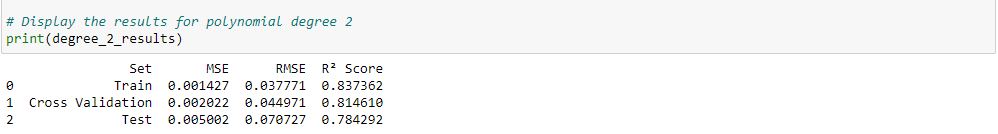
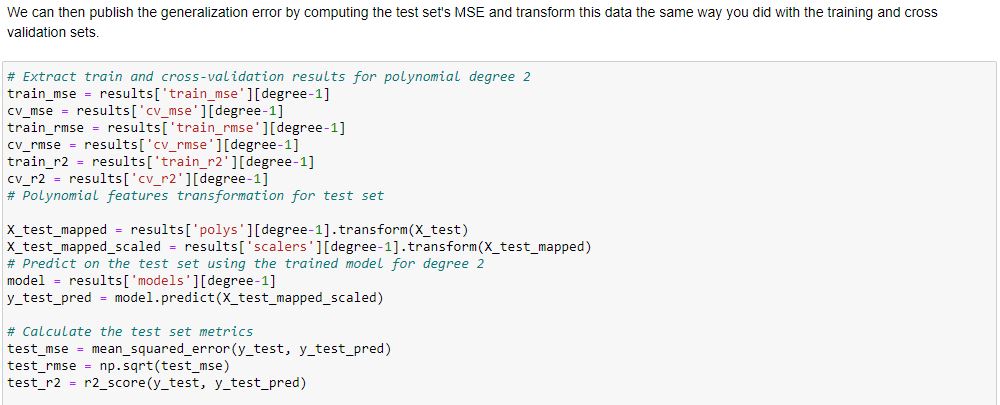
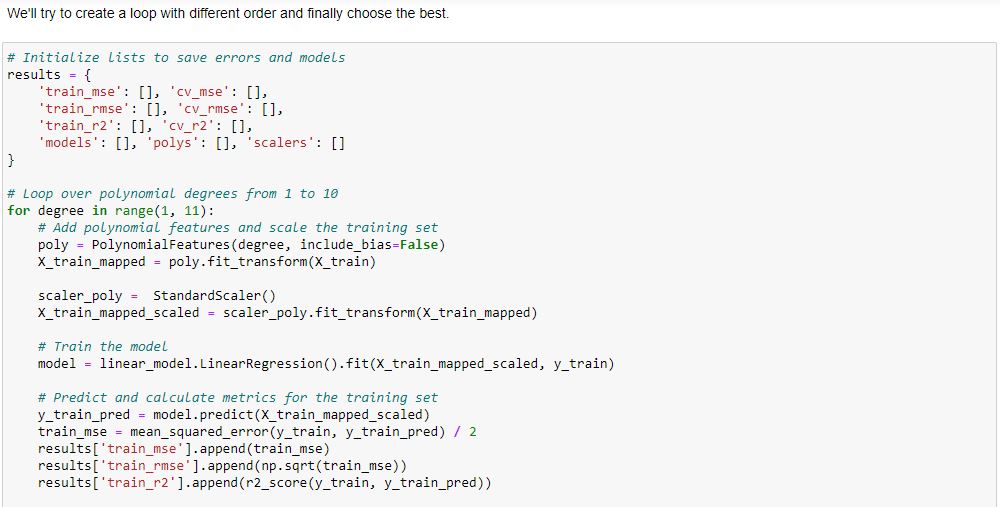
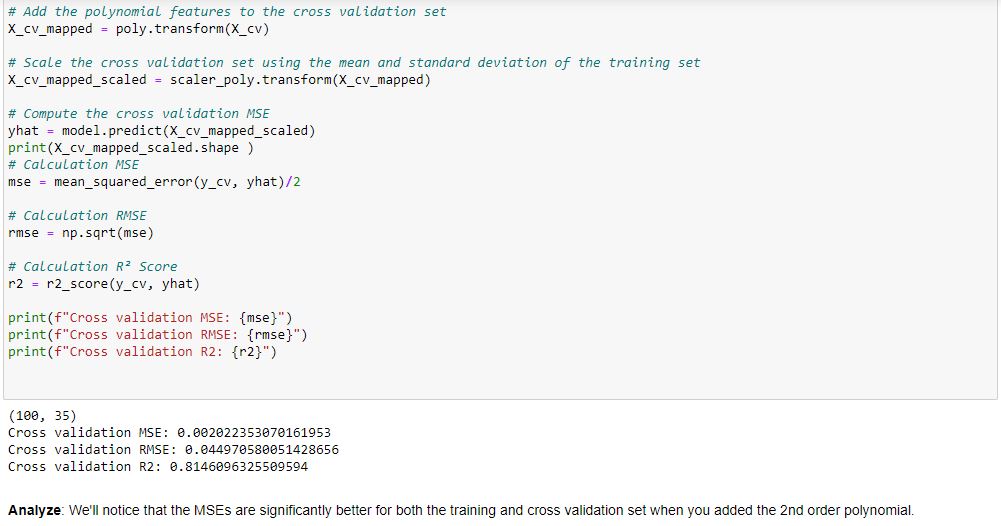
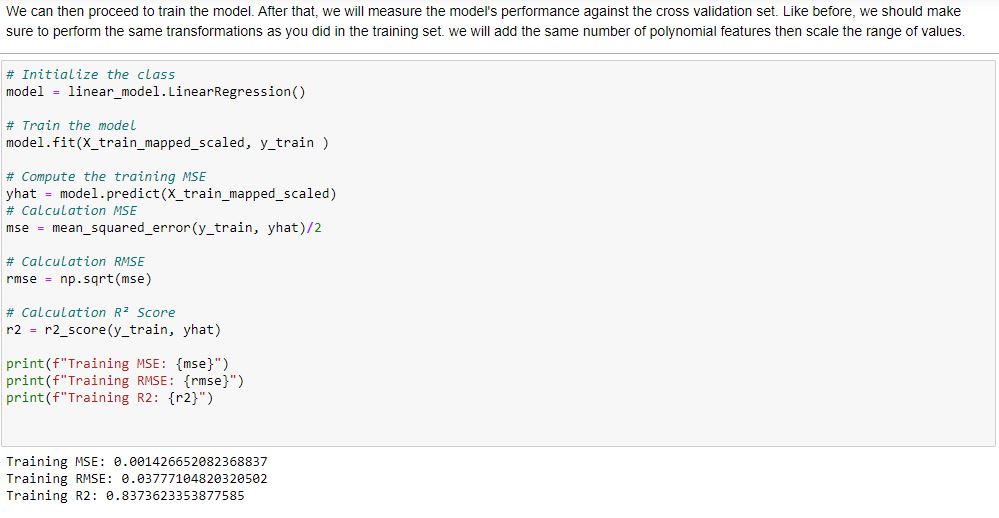
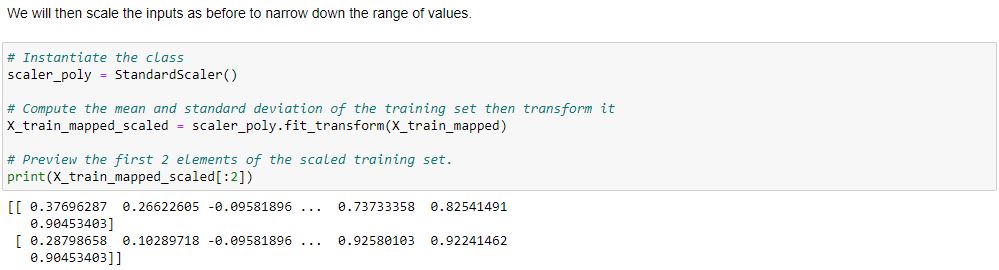
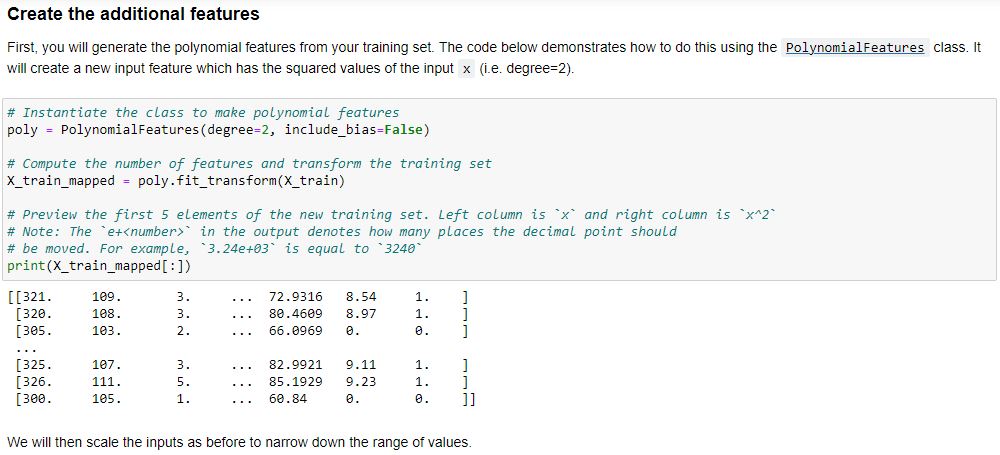
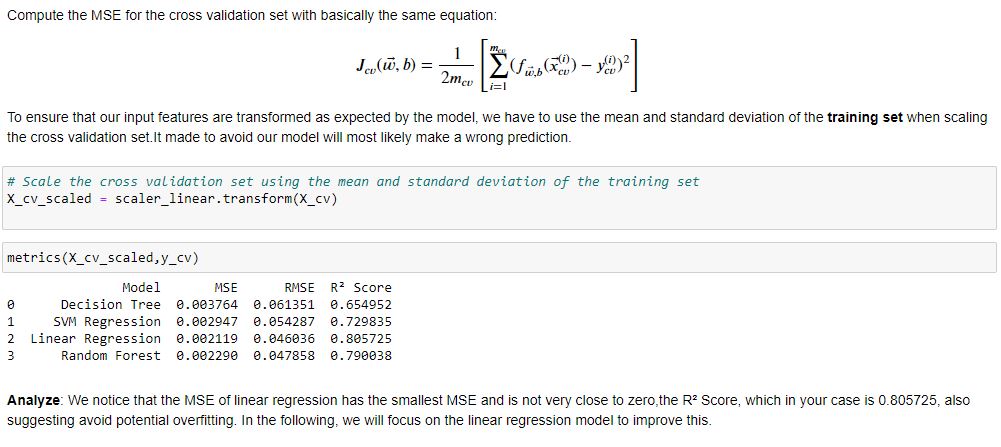
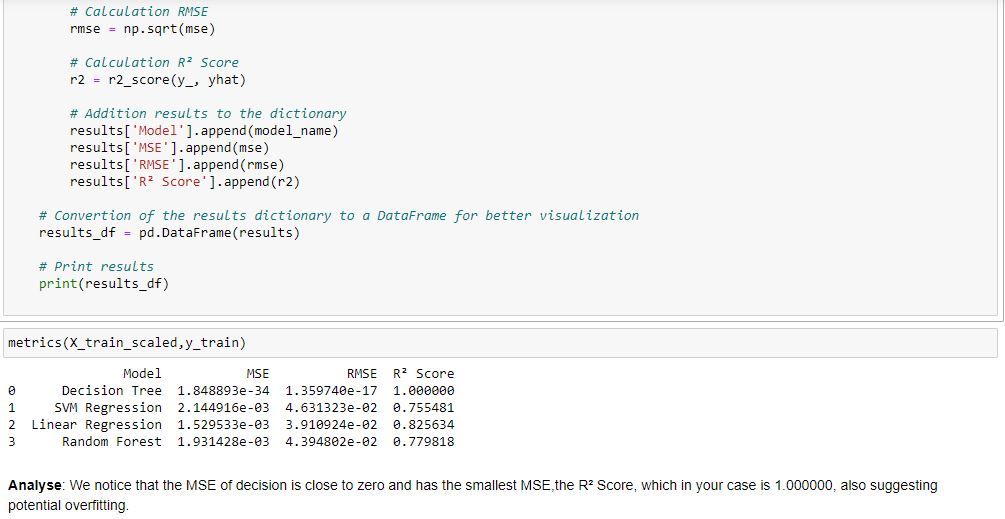
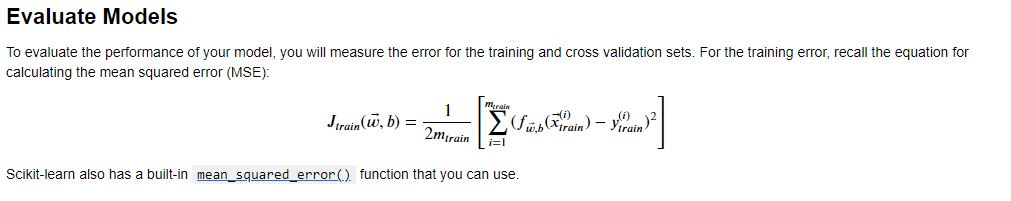
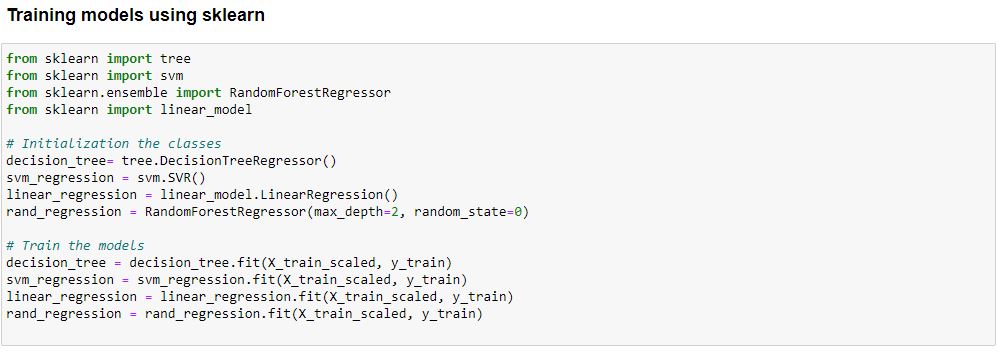
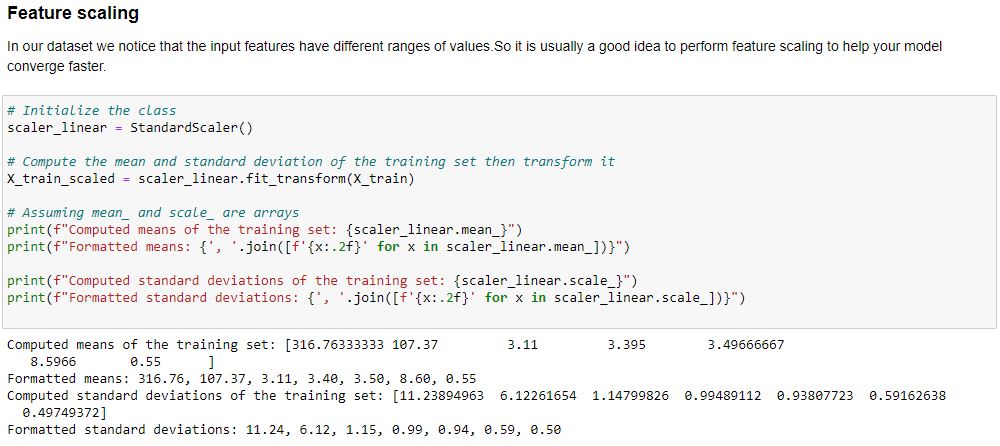
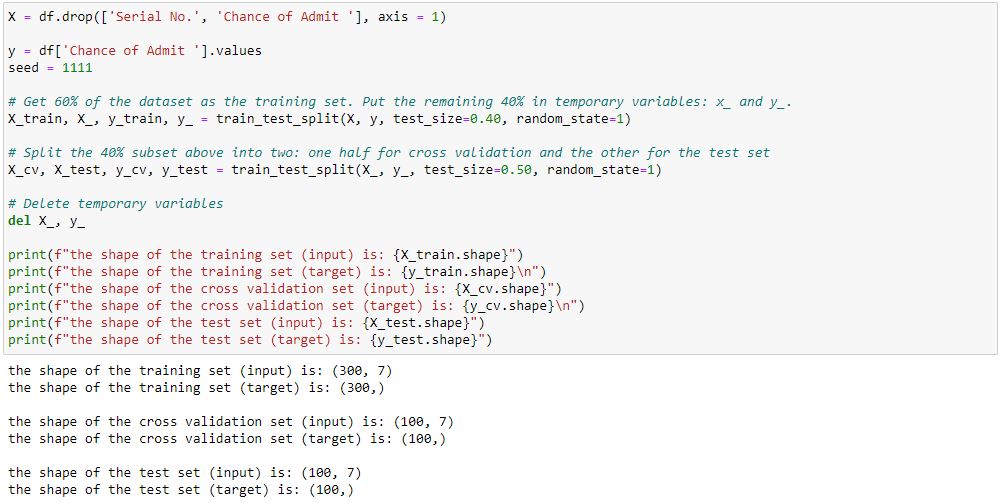
Cross-validation performance is slightly worse than training, but this is expected. The **R² score** has dropped slightly to 0.814, and the **RMSE** has increased, suggesting that the model may not generalize as well as it performs on the training data, but the drop isn't too significant.

**5. Model Deployment**

After loading our model with Pickle, I first want to do a brief test in the Jupyter Notebook. Then, I plan to use FastAPI to build the backend and Streamlit to construct the front end, enabling users to leverage the model. Finally, I will create a Docker image to facilitate the deployment of the application in the cloud.

**6. Code Implementation**

You can find the comments in the pictures.



**Conclusion**

To refine the model further, there are opportunities to improve its accuracy, reliability, and generalization. Techniques such as feature selection, hyperparameter optimization, cross-validation, or ensemble methods can be applied to elevate the model's performance. During the model evaluation process, we compared several machine learning algorithms. Linear Regression was ultimately selected as the best-performing model based on its lowest Mean Squared Error and its ability to generalize well without overfitting. After thoroughly evaluating the training, cross-validation, and test results, the model was deemed ready for deployment. To prepare for deployment with FastAPI, we utilized a scikit-learn pipeline and the pickle library for model serialization.

**References**

 Dataquest. (n.d.). Learning curves in machine learning. Dataquest. Retrieved September 16, 2024, from <https://www.dataquest.io/blog/learning-curves-machine-learning/>

 Sahin, S. (2024, March 5). Scikit-learn pipelines explained: Streamline and optimize your machine learning processes. Medium. <https://medium.com/@sahin.samia/scikit-learn-pipelines-explained-streamline-and-optimize-your-machine-learning-processes-f17b1beb86a4#:~:text=A%20scikit%2Dlearn%20pipeline%20is,cohesively%20as%20a%20composite%20estimator>.

 DATASCIENTEST. (n.d.). FastAPI: Le framework Python parfait pour les APIs. Retrieved from <https://datascientest.com/fastapi>

 Le Data SCIENTIST. (n.d.). Streamlit: Donner une vie à vos modèles de ML. Retrieved from <https://ledatascientist.com/streamlit-donner-une-vie-a-vos-modeles-de-ml/>

 Dataquest. (n.d.). Real-world Python use cases. Retrieved from <https://www.dataquest.io/blog/real-world-python-use-cases/>

 HopHR. (n.d.). Limitations of Jupyter Notebook in data science projects: A step-by-step guide. Retrieved from <https://www.hophr.com/tutorial-page/limitations-jupyter-notebook-data-science-projects-step-by-step-guide#:~:text=For%20instance%2C%20it%20lacks%20robust,large%20datasets%20or%20complex%20computations>.

 Enjoy Algorithms. (n.d.). Introduction to Matplotlib. Retrieved from <https://www.enjoyalgorithms.com/blog/introduction-to-matplotlib>

 Simplilearn. (n.d.). What is FastAPI? Retrieved from <https://www.simplilearn.com/what-is-fastapi-article>

 DATASCIENTEST. (n.d.). Taipy vs Streamlit. Retrieved from <https://datascientest.com/taipy-vs-streamlit>

 Analytics Vidhya. (2023, August). Scikit-learn and TensorFlow. Retrieved from <https://www.analyticsvidhya.com/blog/2023/08/scikit-learn-and-tensorflow/>