[Linear Regression] (CheatSheet)

1. Basic Linear Regression with statsmodels

- **Simple Linear Regression**: `statsmodels.api.OLS(y, X).fit()`
- **Summary of Regression Results**: `results.summary()`
- **Predictions**: `results.predict(X_new)`
- **Residuals**: `results.resid`
- **Regression Plot**: `seaborn.regplot(x, y)`

2. Linear Regression with scikit-learn

- **Fit Linear Model**: `sklearn.linear_model.LinearRegression().fit(X, y)`
- **Coefficients and Intercept**: `model.coef_, model.intercept_`
- **Predictions**: `model.predict(X_new)`
- **R-squared Score**: `model.score(X, y)`
- **Mean Squared Error**: `sklearn.metrics.mean_squared_error(y_true, y_pred)`

3. Data Preprocessing for Linear Regression

- **Standard Scaling**: `sklearn.preprocessing.StandardScaler().fit_transform(X)`
- **Polynomial Features**: `sklearn.preprocessing.PolynomialFeatures(degree).fit_transform(X)`
- **Train-Test Split**: `sklearn.model_selection.train_test_split(X, y)`
- **Handling Missing Values**: `pandas.DataFrame.fillna()`
- **One-Hot Encoding for Categorical Variables**: `pandas.get_dummies()`

4. Diagnostics and Model Checking

- **Plotting Residuals**: `seaborn.residplot(x, y)`
- **Checking for Homoscedasticity**: `statsmodels.stats.diagnostic.het_breuschpagan(residuals, model.model.exog)`
- **Normality Test of Residuals**: `scipy.stats.shapiro(residuals)`

By: Waleed Mousa

- **Outliers Detection (e.g., Cook's distance)**: `statsmodels.stats.outliers_influence.OLSInfluence(model).cooks_distance`
- **Cross-Validation Scores**: `sklearn.model_selection.cross_val_score(model, X, y)`

5. Regularization Techniques

- **Ridge Regression**: `sklearn.linear_model.Ridge(alpha).fit(X, y)`
- **Lasso Regression**: `sklearn.linear_model.Lasso(alpha).fit(X, y)`
- **Elastic Net**: `sklearn.linear_model.ElasticNet(alpha, l1_ratio).fit(X, y)`
- **Grid Search for Hyperparameter Tuning**: `sklearn.model_selection.GridSearchCV()`

6. Multivariate Linear Regression

- **Multiple Linear Regression**: `statsmodels.api.OLS(y, sm.add_constant(X)).fit()`
- **Partial Regression Plots**: `statsmodels.graphics.regressionplots.plot_partregress(y, X, exog_idx)`

7. Advanced Linear Models

- **Generalized Linear Models (GLM)**: `statsmodels.api.GLM(y, X, family).fit()`
- **Quantile Regression**: `statsmodels.regression.quantile_regression.QuantReg(y, X).fit(q)`
- **Robust Regression**: `statsmodels.robust.robust_linear_model.RLM(y, X).fit()`

8. Interaction Effects and Nonlinearity

- **Interaction Terms**: `X['interaction'] = X['feature1'] * X['feature2']`
- **Non-linear Transformations of Predictors**: `numpy.log(X)`, `numpy.sqrt(X)`

9. Model Interpretation

- **Feature Importance**: abs(model.coef_)
- **Coefficients Interpretation**: beta coefficients in results.summary()
- **Effects of Categorical Variables**: one-hot encoded coefficients

10. Model Selection and Evaluation

- **AIC and BIC**: results.aic, results.bic
- **Adjusted R-squared**: 1 - (1 - model.score(X, y)) * ((len(y) - 1) / (len(y) - X.shape[1] - 1))
- **F-Test for Model Significance**: results.f_pvalue
- **Stepwise Regression (Forward, Backward)**: stepwise_selection(X, y) # Custom function

11. Prediction and Confidence Intervals

- **Confidence Interval of Predictions**: results.get_prediction(X_new).conf_int()
- **Prediction Interval**: prediction_interval(model, X_new, alpha) # Custom function

12. Visualization of Linear Models

- **Coefficient Plot**: plot_coefficients(model, feature_names) # Custom function
- **Scatter Plot with Regression Line**: seaborn.lmplot(x, y, data)
- **Partial Dependence Plot**: sklearn.inspection.plot_partial_dependence(model, X, features)

13. Handling Large Datasets

- **Stochastic Gradient Descent for Linear Regression**: sklearn.linear_model.SGDRegressor().fit(X, y)
- **Mini-Batch Gradient Descent**: sklearn.linear_model.SGDRegressor(mini_batch_size)

14. Working with Time Series

- **Linear Regression with Time Series Data**: Handle time-based features and trends in data

- **Lag Features and Autoregression**: `df['lag_feature'] = df['feature'].shift(periods)`

15. Practical Challenges and Solutions

- **Handling Multicollinearity**: Variance Inflation Factor (VIF) calculation
- **Dealing with Non-Stationarity in Time Series**: Differencing or transformation

16. Integrating with Machine Learning Pipelines

- **Using Linear Regression in Pipelines**: `sklearn.pipeline.Pipeline(steps=[('scaler', StandardScaler()), ('regressor', LinearRegression())])`

17. Cross-Validation and Model Selection

- **K-Fold Cross-Validation**: `sklearn.model_selection.cross_val_score(model, X, y, cv=5)`
- **Leave-One-Out Cross-Validation**: `sklearn.model_selection.LeaveOneOut()`
- **Hyperparameter Tuning with GridSearchCV**: `sklearn.model_selection.GridSearchCV(estimator, param_grid)`

18. Diagnostic Plots

- **Residual Plot**: `seaborn.residplot(x, y, lowess=True)`
- **Q-Q Plot for Residuals**: `scipy.stats.probplot(residuals, plot=plt)`
- **Leverage Plot**: `statsmodels.graphics.regressionplots.influence_plot(model, criterion="cooks")`

19. Advanced Feature Engineering

- **Feature Interaction and Polynomial Terms**: `sklearn.preprocessing.PolynomialFeatures(include_bias=False).fit_transform(X)`

- **Automatic Feature Selection**: `sklearn.feature_selection.RFE(estimator, n_features_to_select)`

20. Preprocessing and Feature Scaling

- **Normalization (MinMax Scaling)**: `sklearn.preprocessing.MinMaxScaler().fit_transform(X)`
- **Robust Scaling (handling outliers)**: `sklearn.preprocessing.RobustScaler().fit_transform(X)`

21. Regularization and Penalization Techniques

- **LassoCV for Optimal Alpha**: `sklearn.linear_model.LassoCV(alphas).fit(X, y)`
- **RidgeCV for Optimal Alpha**: `sklearn.linear_model.RidgeCV(alphas).fit(X, y)`
- **ElasticNetCV for Optimal Alpha and L1 Ratio**: `sklearn.linear_model.ElasticNetCV(alphas, l1_ratio).fit(X, y)`

22. Assumptions of Linear Regression

- **Linearity Test**: Plotting observed vs. predicted values
- **Independence Test**: Durbin-Watson test
- **Homoscedasticity Test**: Breusch-Pagan test
- **Normality Test for Residuals**: Kolmogorov-Smirnov test

23. Working with Non-linear Data

- **Transformation of Target Variable**: `numpy.log(y)` or `numpy.sqrt(y)`
- **Generalized Additive Models (GAMs)**: `pygam.LinearGAM().fit(X, y)`

24. Model Interpretability

- **Feature Importance in Linear Models**: `np.abs(model.coef_)`
- **SHAP Values for Linear Regression**: `shap.LinearExplainer(model, X).shap_values(X_new)`

25. Ensemble Methods

- **Averaging Multiple Linear Models**: Averaging predictions from different models
- **Stacking Linear Models**: `sklearn.ensemble.StackingRegressor(estimators)`

26. Error Metrics and Model Evaluation

- **Mean Absolute Error (MAE)**: `sklearn.metrics.mean_absolute_error(y_true, y_pred)`
- **Root Mean Squared Error (RMSE)**: `numpy.sqrt(sklearn.metrics.mean_squared_error(y_true, y_pred))`
- **Mean Squared Logarithmic Error (MSLE)**: `sklearn.metrics.mean_squared_log_error(y_true, y_pred)`

27. Time Series Regression

- **Lag Features for Time Series**: `df['lag_feature'] = df['feature'].shift(1)`
- **Rolling Window Features**: `df['rolling_mean'] = df['feature'].rolling(window=5).mean()`

28. Handling Sparse Data

- **Sparse Matrix Handling**: `scipy.sparse.csr_matrix(X)`
- **Linear Regression with Sparse Data**: `sklearn.linear_model.LinearRegression().fit(X_sparse, y)`

29. Deployment and Persistence of Model

- **Model Serialization with joblib**: `joblib.dump(model, 'model.pkl')`
- **Model Deserialization**: `model = joblib.load('model.pkl')`

30. Performance Improvement

- **Parallel Computing for Large Datasets**: `LinearRegression(n_jobs=-1)`
- **Batch Gradient Descent for Large Datasets**: Implementing batch or mini-batch gradient descent

31. Reporting and Visualization

- **Coefficient Path Plot**: Plotting coefficient magnitude vs. regularization strength
- **Prediction Error Plot**: Yellowbrick's PredictionError(model)

32. Extensions and Related Models

- **Partial Least Squares Regression**: `sklearn.cross_decomposition.PLSRegression()`
- **Ridge Regression with Polynomial Features**: Pipeline with PolynomialFeatures and Ridge

33. Advanced Statistical Techniques

- **Quantile Regression**: `statsmodels.regression.quantile_regression.QuantReg(y, X).fit(q=0.5)`
- **Instrumental Variable Regression**: `linearmodels.iv.IV2SLS(dependent, exog, endog, instruments)`

34. Working with Categorical Variables

- **Encoding and Including Categorical Variables**: `pandas.get_dummies()`
- **ANOVA for Categorical Features Impact**: `statsmodels.api.ols('y ~ C(categorical_feature)', data).fit()`

35. Model Diagnostics and Validation

- **Cross-Validation for Linear Regression**: `sklearn.model_selection.cross_val_score(model, X, y, cv=5)`
- **Learning Curve to Diagnose Model Performance**: `sklearn.model_selection.learning_curve(model, X, y)`

36. Multicollinearity Handling

- **Variance Inflation Factor (VIF) Calculation**: `statsmodels.stats.outliers_influence.variance_inflation_factor(X, i)`

37. Interaction with Domain Knowledge

- **Incorporating Domain Insights into Model**: Modifying features or model based on domain expertise