Vorgehen - Entscheidungsbäume (Code)

Im folgenden wird unser Vorgehen exemplarisch dargestellt. Wegen der Vielzahl an Daten und der damit verbundenen hohen Rechenzeit haben wir aus den hier vorgestellten Ansätzen Skripte zum ausführen erstellt, damit unsere eigenen Rechner und Entwicklungsumgebungen nicht von der langen Rechnung beeinflusst werden. Zu Auswertung haben wir uns diverse Plots und Tabellen als csv Dateien ausgeben lassen. Odnernamen und sonstige allgemeine relevante Titel haben wir in der Datei "Global_configurations.py" gespeichert.

Datenerhebung

Durch unseren Ansatz mit möglich vielen Daten zu arbeiten haben wir versucht so viele Datensätze wie möglich herunterzuladen und als Features zu verwenden. Für die Weltbankdaten wird der Massendownload verwendet und die Daten werden vorbereitet. Nachvollziehbarkeit: "prepare_WB.py" Das Vorgehen für die OECD:

1. Extrahieren aller Keys für Datensätzer die mit der API verfügbar sind aus dem Quellcode der Seite
2. Herunterladen aller verfügbaren Datensätze für den Zeitraum von 1998-2018 und speichern als csv.

In [2]:
```
from Global_configuartions import *
import requests as rq
import re
from functions_downloader import oecd_data_load_and_save
from tqdm import tqdm

# get all the keys
r = rq.get('https://stats.oecd.org/')
# set to remove duplicates and then list to make it easier to handle
keys = sorted(list(set(re.findall(re.compile(r'\?DataSet=(.*?)"'), r.text)
)))
with open("../keys.csv", "w", encoding='utf-8') as file:
    file.write('\n'.join(keys))
keys
```

Out[2]:
```
['7HA_A_Q',
 '7IA_A_Q',
 '7II_INDIC',
 '7PSD_D1D4',
 'AEA',
 'AEI_NUTRIENTS',
 'AEI_OTHER',
 'AFA_CALC_IN3',
 'AFA_CALC_OUT3',
 'AFA_IN3',
 'AFA_IN3_MANUF',
 'AFA_OUT3',
 'AFA_OUT3_PARTNER',
 'AFDDANN',
```

```
                'AFRICAPOLIS',
                'AGE_GAP',
                'AIRTRANS_CO2',
                'AIR_EMISSIONS',
                'AIR_GHG',
                'ALFS_EMP',
                'ALFS_EMP_ARCHIVE',
                'ALFS_POP_LABOUR',
                'ALFS_POP_VITAL',
                'ALFS_SUMTAB',
                'ALFS_SUMTAB_ARCHIVE',
                'AMNE_IN',
                'AMNE_IN_PARTNER',
                'AMNE_OUT',
                'AMNE_OUT_PARTNER',
                'ANBERD2011_REV3',
                'ANBERD_REV2',
                'ANBERD_REV4',
                'ANHRS',
                'AVD_DUR',
                'AVE_HRS',
                'AV_AN_WAGE',
                'AWCOMP',
                'AWCOU',
                'BATIS_EBOPS2002',
                'BATIS_EBOPS2010',
                'BENCHMARK_STIO',
                'BERD_COST',
                'BERD_COST_ISIC4',
                'BERD_FUNDS',
                'BERD_FUNDS_ISIC4',
                'BERD_INDU',
                'BERD_INDUSTRY',
                'BERD_INDUSTRY_ISIC4',
                'BERD_MA_SOF',
                'BERD_MA_TOE',
                'BERD_SIZE',
                'BERD_SOF_SIZE',
                'BERD_STIO',
                'BFGID',
                'BFSIGI',
                'BIMTS_CPA',
                'BLI',
                'BLI2013',
                'BLI2014',
                'BLI2015',
                'BLI2016',
                'BLI2017',
                'BPF1',
                'BROADBAND_DB',
                'BSI',
                'BTDIXE',
                'BTDIXE_I3',
                'BTDIXE_I4',
                'BTDIXE_ISIC3_2011',
                'BTD_ED_2010',
                'BUILT_UP',
```

```
                'BUILT_UP_FUA',
                'CALCULATED',
                'CBC',
                'CBCR_TABLEI',
                'CBCR_TABLEII',
                'CBCR_TABLEIII',
                'CBCR_TABLEIV',
                'CBCR_TABLEV',
                'CHAPTER_A_EAG2014_NEW',
                'CHAPTER_B_EAG2014',
                'CHAPTER_C_EAG2014',
                'CHAPTER_D_EAG2014',
                'CIF_FOB_ITIC',
                'CIIE10',
                'CITIES',
                'CPA',
                'CPL',
                'CPSE_2010',
                'CRS1',
                'CRS1_GREQ',
                'CS',
                'CSE_2010',
                'CSPCUBE',
                'CTS_CIT',
                'CTS_ETR',
                'CTS_REV',
                'DACDEFL',
                'DACGEO',
                'DACIND',
                'DACSECTOR',
                'DECOMP',
                'DEC_I',
                'DIOC_CITIZEN_AGE',
                'DIOC_DURATION_STAY',
                'DIOC_FIELD_STUDY',
                'DIOC_LFS',
                'DIOC_OCCUPATION',
                'DIOC_OCCUPATION_DET',
                'DIOC_SECTOR',
                'DIOC_SEX_AGE',
                'DUR_D',
                'DUR_I',
                'DV_DCD_GENDER',
                'DV_DCD_PPFD',
                'EAG_AL',
                'EAG_EARNINGS',
                'EAG_EA_SKILLS',
                'EAG_ENRL_MOBILES_FIELDS',
                'EAG_ENRL_MOBILES_ORIGIN',
                'EAG_ENRL_RATE_AGE',
                'EAG_ENRL_SHARE_CATEGORY',
                'EAG_ENRL_SHARE_INST',
                'EAG_ESO',
                'EAG_FIN_ANNEX2',
                'EAG_FIN_NATURE',
                'EAG_FIN_RATIO',
                'EAG_FIN_SOURCE',
```

```
                'EAG_FIN_STUDENTS',
                'EAG_GRAD_ENTR_FIELD',
                'EAG_GRAD_ENTR_RATES',
                'EAG_GRAD_ENTR_SHARE',
                'EAG_IT_ALL',
                'EAG_MIGR',
                'EAG_MOB',
                'EAG_NEAC',
                'EAG_PERS_RATIO',
                'EAG_PERS_SHARE_AGE',
                'EAG_TRANS',
                'EAG_TS_ACT',
                'EAG_TS_STA',
                'EAG_WT_ORG',
                'EAMFP',
                'EAR_MEI',
                'ECONSH_D',
                'ECONSH_I',
                'EDU_CLASS',
                'EDU_DEM',
                'EDU_ENRL_AGE',
                'EDU_ENRL_FIELD',
                'EDU_ENRL_INST',
                'EDU_ENRL_MOBILE',
                'EDU_ENTR_AGE',
                'EDU_ENTR_FIELD',
                'EDU_FIN_SUBNAT',
                'EDU_GRAD_AGE',
                'EDU_GRAD_FIELD',
                'EDU_GRAD_MOBILE',
                'EDU_PERS_AGE',
                'EDU_PERS_INST',
                'EDU_PERS_MANA',
                'EGDNA_PUBLIC',
                'EGDNA_PUBLIC_SOCIO',
                'ELSPENSIONS',
                'ENVPERFINDIC_TAD_2008',
                'EO',
                'EO01_VINTAGE',
                'EO02_VINTAGE',
                'EO03_VINTAGE',
                'EO04_VINTAGE',
                'EO05_VINTAGE',
                'EO06_VINTAGE',
                'EO07_VINTAGE',
                'EO08_VINTAGE',
                'EO09_VINTAGE',
                'EO100_INTERNET',
                'EO101_INTERNET',
                'EO102_INTERNET',
                'EO103_INTERNET',
                'EO103_LTB',
                'EO104_INTERNET',
                'EO105_INTERNET',
                'EO106_INTERNET',
                'EO107_INTERNET_1',
                'EO107_INTERNET_2',
```

```
                'EO108_INTERNET',
                'EO10_VINTAGE',
                'EO11_VINTAGE',
                'EO12_VINTAGE',
                'EO13_VINTAGE',
                'EO14_VINTAGE',
                'EO15_VINTAGE',
                'EO16_VINTAGE',
                'EO17_VINTAGE',
                'EO18_VINTAGE',
                'EO19_VINTAGE',
                'EO20_VINTAGE',
                'EO21_VINTAGE',
                'EO22_VINTAGE',
                'EO23_VINTAGE',
                'EO24_VINTAGE',
                'EO25_VINTAGE',
                'EO26_VINTAGE',
                'EO27_VINTAGE',
                'EO28_VINTAGE',
                'EO29_VINTAGE',
                'EO30_VINTAGE',
                'EO31_VINTAGE',
                'EO32_VINTAGE',
                'EO33_VINTAGE',
                'EO34_VINTAGE',
                'EO35_VINTAGE',
                'EO36_VINTAGE',
                'EO37_VINTAGE',
                'EO38_VINTAGE',
                'EO39_VINTAGE',
                'EO40_VINTAGE',
                'EO41_VINTAGE',
                'EO42_VINTAGE',
                'EO43_VINTAGE',
                'EO44_VINTAGE',
                'EO45_VINTAGE',
                'EO46_VINTAGE',
                'EO47_VINTAGE',
                'EO48_VINTAGE',
                'EO49_VINTAGE',
                'EO50_VINTAGE',
                'EO51_VINTAGE',
                'EO52_VINTAGE',
                'EO53_VINTAGE',
                'EO54_VINTAGE',
                'EO55_VINTAGE',
                'EO56_VINTAGE',
                'EO57_VINTAGE',
                'EO58_VINTAGE',
                'EO59_VINTAGE',
                'EO60_MAIN',
                'EO60_TRADE',
                'EO61_MAIN',
                'EO61_TRADE',
                'EO62_MAIN',
                'EO62_TRADE',
```

```
                'EO63_MAIN',
                'EO63_TRADE',
                'EO64_MAIN',
                'EO64_TRADE',
                'EO65_MAIN',
                'EO65_TRADE',
                'EO66_MAIN',
                'EO66_TRADE',
                'EO67_MAIN',
                'EO67_TRADE',
                'EO68_MAIN',
                'EO68_TRADE',
                'EO69_MAIN',
                'EO69_TRADE',
                'EO70_MAIN',
                'EO70_TRADE',
                'EO71_MAIN',
                'EO71_TRADE',
                'EO72_MAIN',
                'EO72_TRADE',
                'EO73_MAIN',
                'EO73_TRADE',
                'EO74_MAIN',
                'EO74_TRADE',
                'EO75_MAIN',
                'EO75_TRADE',
                'EO76_MAIN',
                'EO76_TRADE',
                'EO77_MAIN',
                'EO77_TRADE',
                'EO78_MAIN',
                'EO78_TRADE',
                'EO79_MAIN',
                'EO79_TRADE',
                'EO80_MAIN',
                'EO80_TRADE',
                'EO81_MAIN',
                'EO81_TRADE',
                'EO82_MAIN',
                'EO82_TRADE',
                'EO83_FLASHFILE_EO83',
                'EO83_MAIN',
                'EO83_TRADE',
                'EO84_FLASHFILE_EO84',
                'EO84_MAIN',
                'EO84_TRADE',
                'EO85_FLASHFILE_EO85',
                'EO85_MAIN',
                'EO85_TRADE',
                'EO86_FLASHFILE_EO86',
                'EO86_MAIN',
                'EO86_TRADE',
                'EO87_OUTLOOK87',
                'EO88_FLASHFILE_EO88',
                'EO88_INTERNET',
                'EO89_FLASHFILE_EO89',
                'EO89_INTERNET',
```

```
'EO90_FLASHFILE_EO90',
'EO90_INTERNET',
'EO91_FLASHFILE_EO91',
'EO91_INTERNET',
'EO91_LTB',
'EO92_FLASHFILE_EO92',
'EO92_INTERNET',
'EO93_FLASHFILE_EO93',
'EO93_INTERNET',
'EO93_LTB',
'EO94_INTERNET',
'EO95_INTERNET',
'EO95_LTB',
'EO96_INTERNET',
'EO97_INTERNET',
'EO97_OUTLOOK97',
'EO98_INTERNET',
'EO99_INTERNET',
'EPEA',
'EPER',
'EPL_CD',
'EPL_OV',
'EPL_R',
'EPL_T',
'EPS',
'ERTR',
'ERTR_ACC',
'ETCR',
'EXP_COFOG_SPECIAL',
'EXP_MORSC',
'EXP_PM2_5',
'FACTBOOK2014_PUB',
'FACTBOOK2015_PUB',
'FAMILY',
'FDIINDEX',
'FDI_AGGR_SUMM',
'FDI_BOP_IIP',
'FDI_CTRY_ECO_HIST',
'FDI_CTRY_IND_SUMM',
'FDI_FLOW_AGGR',
'FDI_FLOW_CTRY',
'FDI_FLOW_IND',
'FDI_FLOW_INDUSTRY',
'FDI_FLOW_PARTNER',
'FDI_INC_AGGR',
'FDI_INC_CTRY',
'FDI_INC_IND',
'FDI_POSITION_INDUSTRY',
'FDI_POSITION_PARTNER',
'FDI_POS_AGGR',
'FDI_POS_CTRY',
'FDI_POS_IND',
'FDI_POS_IND_TEMP_A',
'FDI_POS_PARTNER_TEMP_A',
'FFS_ARG',
'FFS_AUS',
'FFS_AUT',
```

```
                'FFS_BEL',
                'FFS_BRA',
                'FFS_CAN',
                'FFS_CHE',
                'FFS_CHL',
                'FFS_CHN',
                'FFS_COL',
                'FFS_CZE',
                'FFS_DEU',
                'FFS_DNK',
                'FFS_ESP',
                'FFS_EST',
                'FFS_FIN',
                'FFS_FRA',
                'FFS_GBR',
                'FFS_GRC',
                'FFS_HUN',
                'FFS_IDN',
                'FFS_IND',
                'FFS_INDICATOR_DETAILED',
                'FFS_IRL',
                'FFS_ISL',
                'FFS_ISR',
                'FFS_ITA',
                'FFS_JPN',
                'FFS_KOR',
                'FFS_LTU',
                'FFS_LUX',
                'FFS_LVA',
                'FFS_MEX',
                'FFS_NLD',
                'FFS_NOR',
                'FFS_NZL',
                'FFS_POL',
                'FFS_PRT',
                'FFS_RUS',
                'FFS_SVK',
                'FFS_SVN',
                'FFS_SWE',
                'FFS_TUR',
                'FFS_USA',
                'FFS_ZAF',
                'FIN_IND_FA',
                'FIN_IND_FBS',
                'FISH_AQUA',
                'FISH_EMPL',
                'FISH_FLD',
                'FISH_FLEET',
                'FISH_FSE',
                'FISH_GFT',
                'FISH_INLAND',
                'FISH_LAND',
                'FISH_PAT_COL',
                'FISH_PAT_COL_RATES',
                'FISH_PAT_DEV',
                'FISH_PAT_DIFF',
                'FISH_PAT_RD',
```

```
                'FISH_RECRE',
                'FISH_TRADE',
                'FOBS',
                'FOOD_WASTE',
                'FOREST',
                'FSS',
                'FTPTC_D',
                'FTPTC_I',
                'FTPTN_D',
                'FTPTN_I',
                'G20_PRICES',
                'GBAORD_NABS2007',
                'GBARD_NABS2007',
                'GENDER_EMP',
                'GENDER_ENT1',
                'GERD_COST',
                'GERD_FORD',
                'GERD_FUNDS',
                'GERD_FUNDS_PRE1981',
                'GERD_OBJECTIVE_NABS2007',
                'GERD_SCIENCE',
                'GERD_SEO',
                'GERD_SOF',
                'GERD_TOE',
                'GERD_TORD',
                'GFG',
                'GID2',
                'GIDDB2012',
                'GIDDB2014',
                'GIDDB2019',
                'GOV',
                'GOV_2011',
                'GOV_2013',
                'GOV_2015',
                'GOV_2017',
                'GOV_2019',
                'GOV_DEBT',
                'GOV_LAC',
                'GOV_SEA',
                'GOV_WB',
                'GREEN_GROWTH',
                'GRR',
                'GSSE_2010',
                'GVC_INDICATORS',
                'G_FUNCTION_AMOUNTS',
                'G_FUNCTION_RATIO',
                'G_TYPE_AMOUNTS',
                'G_TYPE_RATIO',
                'HEALTH_DEMR',
                'HEALTH_ECOR',
                'HEALTH_HCQI',
                'HEALTH_LTCR',
                'HEALTH_LVNG',
                'HEALTH_PHMC',
                'HEALTH_PROC',
                'HEALTH_PROT',
                'HEALTH_REAC',
```

```
                'HEALTH_STAT',
                'HEALTH_WFMI',
                'HGRR',
                'HH_DASH',
                'HIGH_AGLINK_2009',
                'HIGH_AGLINK_2010',
                'HIGH_AGLINK_2011',
                'HIGH_AGLINK_2012',
                'HIGH_AGLINK_2013',
                'HIGH_AGLINK_2014',
                'HIGH_AGLINK_2015',
                'HIGH_AGLINK_2016',
                'HIGH_AGLINK_2017',
                'HIGH_AGLINK_2018',
                'HIGH_AGLINK_2019',
                'HIGH_AGLINK_2020',
                'HISTPOP',
                'HOURSPOV',
                'HOUSE_PRICES',
                'HSL',
                'IA',
                'ICT_BUS',
                'ICT_HH2',
                'IDD',
                'IMW',
                'INDICE_OECD',
                'INSIND',
                'INTACT_FOREST_LANDSCAPES',
                'INVPT_D',
                'INVPT_I',
                'IOTS',
                'IOTSI4_2018',
                'IO_GHG_2015',
                'IO_GHG_2019',
                'IPM_STIO',
                'IRTAD_CASUAL_BY_AGE',
                'ITF_ACCESS',
                'ITF_CQ',
                'ITF_GOODS_TRANSPORT',
                'ITF_INDICATORS',
                'ITF_INV-MTN_DATA',
                'ITF_PASSENGER_TRANSPORT',
                'ITF_ROAD_ACCIDENTS',
                'ITF_ROAD_HAULAGE',
                'ITF_SHORT_TERM_INDIC',
                'JOBQ',
                'JOBQ_I',
                'KEI',
                'KEY_STIO',
                'LAB_REG_VAC',
                'LAC_REVCHL',
                'LAC_REVCOL',
                'LAC_REVMEX',
                'LAND_COVER',
                'LAND_COVER_CHANGE',
                'LAND_COVER_FUA',
                'LAND_USE',
```

```
                'LEVEL',
                'LFS_D',
                'LFS_SEXAGE_I_C',
                'LFS_SEXAGE_I_R',
                'LMPEXP',
                'MATERIAL_RESOURCES',
                'MA_D',
                'MA_I',
                'MEI',
                'MEI_ARCHIVE',
                'MEI_BOP',
                'MEI_BOP6',
                'MEI_BTS_COS',
                'MEI_CLI',
                'MEI_CTRY_WEIGHTS',
                'MEI_FIN',
                'MEI_PRICES_PPI',
                'MEI_REAL',
                'MEI_TRD',
                'MEM4',
                'MEMTAX',
                'METR',
                'MFP',
                'MIG_EMP_EDUCATION',
                'MIG_NUP_RATES_GENDER',
                'MIN2AVE',
                'MISMATCH',
                'MON20113_1',
                'MON20113_1EE',
                'MON20113_2',
                'MON20113_2EE',
                'MON20113_3',
                'MON20113_3EE',
                'MON20113_4',
                'MON20113_4EE',
                'MON20113_5',
                'MON20113_5EE',
                'MON20113_6',
                'MON20113_6EE',
                'MON20113_7',
                'MON20113_7EE',
                'MON2011AANRI',
                'MON2011AANRI_EE',
                'MON2011CSCT',
                'MON2011CSCT_EE',
                'MON2011CSCT_O',
                'MON2011PSCT',
                'MON2011PSCT_EE',
                'MON2011PSCT_O',
                'MON2011TSE',
                'MON2011TSE_EE',
                'MON2011TSE_O',
                'MON20123_1',
                'MON20123_1EE',
                'MON20123_2',
                'MON20123_2EE',
                'MON20123_3',
```

```
'MON20123_3EE',
'MON20123_4',
'MON20123_4EE',
'MON20123_5',
'MON20123_5EE',
'MON20123_6',
'MON20123_6EE',
'MON20123_7',
'MON20123_7EE',
'MON2012AANRI',
'MON2012AANRI_EE',
'MON2012CSCT',
'MON2012CSCT_EE',
'MON2012CSCT_O',
'MON2012PSCT',
'MON2012PSCT_EE',
'MON2012PSCT_O',
'MON2012TSE',
'MON2012TSE_EE',
'MON2012TSE_O',
'MON2013_REFERENCE_TABLE',
'MON2014_REFERENCE_TABLE',
'MON2015_REFERENCE_TABLE',
'MON2016_REFERENCE_TABLE',
'MON2017_REFERENCE_TABLE',
'MON2017_SINGLE_COMMODITY_INDICATORS',
'MON2018_REFERENCE_TABLE',
'MON2018_SINGLE_COMMODITY_INDICATORS',
'MON2019_REFERENCE_TABLE',
'MON2019_SINGLE_COMMODITY_INDICATORS',
'MON_REFERENCE_TABLE',
'MON_SINGLE_COMMODITY_INDICATORS',
'MSTI_PUB',
'MTC',
'MTPI_CPA',
'MULTISYSTEM',
'MUNW',
'MW_CURP',
'NAAG',
'NAAG_2009',
'NAAG_2010',
'NAAG_2011',
'NAAG_2013',
'NAAG_2014',
'NAAG_2015_NOV15',
'NAT_RES',
'NCC',
'NRR',
'OCEAN',
'OECD-AEA',
'OECD_TSE2010',
'ONRD_COST',
'ONRD_FUNDS',
'PAG',
'PART2',
'PARTNER',
'PATS_COOP',
```

```
                'PATS_IPC',
                'PATS_REGION',
                'PAT_COL',
                'PAT_COL_RATES',
                'PAT_DEV',
                'PAT_DIFF',
                'PAT_IND',
                'PDBI',
                'PDBI_I4',
                'PDB_GR',
                'PDB_LV',
                'PDYGTH',
                'PERS_FORD',
                'PERS_FUNC',
                'PERS_INDU',
                'PERS_INDUSTRY',
                'PERS_INDUSTRY_ISIC4',
                'PERS_OCCUP',
                'PERS_OCCUP_PRE1981',
                'PERS_QUAL',
                'PERS_QUALIF',
                'PERS_SCIENCE',
                'PMR',
                'PMR2018',
                'PNNI_NEW',
                'PNN_NEW',
                'POP_FIVE_HIST',
                'POP_PROJ',
                'PPP2005',
                'PPP2008',
                'PPP2011',
                'PPP2014',
                'PPP2017',
                'PPPGDP',
                'PPRF',
                'PRICES_CPI',
                'PROFSVC',
                'PROTECTED_AREAS',
                'PSE_2010',
                'PT1',
                'PT2',
                'PT3',
                'PT4',
                'PT5',
                'PT6',
                'PT7',
                'PT8',
                'PT9',
                'PTRCCSA',
                'PTRCCUB',
                'PTRSA',
                'PTRUB',
                'QASA_7HH',
                'QASA_7II',
                'QASA_7II_DISCR',
                'QASA_7II_INDIC',
                'QASA_TABLE610',
```

```
                'QASA_TABLE610R',
                'QASA_TABLE620',
                'QASA_TABLE620R',
                'QASA_TABLE625R',
                'QASA_TABLE710',
                'QASA_TABLE710R',
                'QASA_TABLE720',
                'QASA_TABLE720R',
                'QASA_TABLE725R',
                'QASA_TABLE7PSD',
                'QASA_TABLE801',
                'QASA_TABLE801_ARCHIVE',
                'QITS',
                'QNA',
                'QNA_ARCHIVE',
                'RDSUB',
                'RDTAX',
                'RD_ACTIVITY',
                'RD_ACTIVITY_PRE1981',
                'REFSERIES_ASI',
                'REFSERIES_GL',
                'REFSERIES_LAC',
                'REFSERIES_MSIT',
                'REFSERIES_REV',
                'REF_TOTALOFFICIAL',
                'REF_TOTALRECPTS',
                'REF_TOTAL_ODF',
                'REGION_DEMOGR',
                'REGION_ECONOM',
                'REGION_EDUCAT',
                'REGION_INNOVATION',
                'REGION_LABOUR',
                'REGION_MIGRANTS',
                'REGION_SOCIAL',
                'REGION_TYPOL',
                'REG_BUSI_DEMOG',
                'RETAIL',
                'REV',
                'REVARG',
                'REVAUS',
                'REVAUS_AP',
                'REVAUT',
                'REVBAR',
                'REVBEL',
                'REVBFA',
                'REVBGR',
                'REVBHS',
                'REVBLZ',
                'REVBOL',
                'REVBRA',
                'REVBTN',
                'REVCAN',
                'REVCHE',
                'REVCHL',
                'REVCHN',
                'REVCIV',
                'REVCMR',
```

```
                'REVCOD',
                'REVCOG',
                'REVCOK',
                'REVCOL',
                'REVCPV',
                'REVCRI',
                'REVCUB',
                'REVCZE',
                'REVDEU',
                'REVDNK',
                'REVDOM',
                'REVECU',
                'REVEGY',
                'REVENUE_OUT',
                'REVESP',
                'REVEST',
                'REVEU',
                'REVFIN',
                'REVFJI',
                'REVFRA',
                'REVGBR',
                'REVGHA',
                'REVGRC',
                'REVGTM',
                'REVGUY',
                'REVHON',
                'REVHUN',
                'REVIDN',
                'REVIRL',
                'REVISL',
                'REVISR',
                'REVITA',
                'REVJAM',
                'REVJPN',
                'REVJPN_ASIAN',
                'REVKAZ',
                'REVKEN',
                'REVKOR',
                'REVKOR_ASIAN',
                'REVLCA',
                'REVLIE',
                'REVLSO',
                'REVLTU',
                'REVLUX',
                'REVLVA',
                'REVMAR',
                'REVMDG',
                'REVMEX',
                'REVMLI',
                'REVMNG',
                'REVMUS',
                'REVMYS',
                'REVNAM',
                'REVNER',
                'REVNGA',
                'REVNIC',
                'REVNLD',
```

```
                'REVNOR',
                'REVNRU',
                'REVNZL',
                'REVNZL_AP',
                'REVPAN',
                'REVPER',
                'REVPHL',
                'REVPNG',
                'REVPOL',
                'REVPRT',
                'REVPRY',
                'REVRWA',
                'REVSEN',
                'REVSGP',
                'REVSLB',
                'REVSLV',
                'REVSVK',
                'REVSVN',
                'REVSWE',
                'REVSWZ',
                'REVSYC',
                'REVTCD',
                'REVTGO',
                'REVTHA',
                'REVTKL',
                'REVTTO',
                'REVTUN',
                'REVTUR',
                'REVUGA',
                'REVURY',
                'REVUSA',
                'REVVEN',
                'REVVUT',
                'REVWSM',
                'REVZAF',
                'RE_FIT',
                'RFD',
                'RFIN1',
                'RHPI',
                'RHPI_TARGET',
                'RIOMARKERS',
                'RMW',
                'RPERS',
                'RSLACT',
                'RS_AFR',
                'RS_ASI',
                'RS_GBL',
                'RTA_STIO',
                'RWB',
                'SBE',
                'SDBS_BDI',
                'SDBS_BDI_ISIC4',
                'SECTREG2018',
                'SHA',
                'SHA_FP',
                'SHA_FS',
                'SHA_HK',
```

```
                'SIGI2014',
                'SIGI2019',
                'SITC_SECTION',
                'SKILLS_2018_INDUSTRY',
                'SKILLS_2018_REGION',
                'SKILLS_2018_TOTAL',
                'SMES_SCOREBOARD',
                'SNA_TABLE1',
                'SNA_TABLE10',
                'SNA_TABLE10_ARCHIVE',
                'SNA_TABLE10_SNA93',
                'SNA_TABLE11',
                'SNA_TABLE11_ARCHIVE',
                'SNA_TABLE11_SNA93',
                'SNA_TABLE12',
                'SNA_TABLE12_ARCHIVE',
                'SNA_TABLE12_SNA93',
                'SNA_TABLE13',
                'SNA_TABLE13_ARCHIVE',
                'SNA_TABLE13_SNA93',
                'SNA_TABLE14A',
                'SNA_TABLE14A_ARCHIVE',
                'SNA_TABLE14A_SNA93',
                'SNA_TABLE1_ARCHIVE',
                'SNA_TABLE1_SNA93',
                'SNA_TABLE2',
                'SNA_TABLE29',
                'SNA_TABLE2_ARCHIVE',
                'SNA_TABLE2_SNA93',
                'SNA_TABLE3',
                'SNA_TABLE30',
                'SNA_TABLE31',
                'SNA_TABLE3_ARCHIVE',
                'SNA_TABLE3_SNA93',
                'SNA_TABLE4',
                'SNA_TABLE40',
                'SNA_TABLE41',
                'SNA_TABLE42',
                'SNA_TABLE43',
                'SNA_TABLE44',
                'SNA_TABLE45',
                'SNA_TABLE5',
                'SNA_TABLE5_ARCHIVE',
                'SNA_TABLE5_SNA93',
                'SNA_TABLE6',
                'SNA_TABLE610',
                'SNA_TABLE610R',
                'SNA_TABLE620',
                'SNA_TABLE620R',
                'SNA_TABLE625R',
                'SNA_TABLE6A',
                'SNA_TABLE6A_ARCHIVE',
                'SNA_TABLE6A_SNA93',
                'SNA_TABLE6_SNA93',
                'SNA_TABLE7',
                'SNA_TABLE710',
                'SNA_TABLE710R',
```

```
'SNA_TABLE720',
'SNA_TABLE720R',
'SNA_TABLE725R',
'SNA_TABLE750',
'SNA_TABLE7A',
'SNA_TABLE7A_ARCHIVE',
'SNA_TABLE7A_SNA93',
'SNA_TABLE7_SNA93',
'SNA_TABLE8',
'SNA_TABLE8A',
'SNA_TABLE8A_ARCHIVE',
'SNA_TABLE8A_SNA93',
'SNA_TABLE8_SNA93',
'SNA_TABLE9',
'SNA_TABLE9A',
'SNA_TABLE9A_ARCHIVE',
'SNA_TABLE9A_SNA93',
'SNA_TABLE9B',
'SNA_TABLE9B_ARCHIVE',
'SNA_TABLE9B_SNA93',
'SNA_TABLE9_SNA93',
'SNGF',
'SOCR',
'SOCR_REF',
'SOCX_AGG',
'SOCX_DET',
'SOCX_REF',
'SSIS_BSC',
'SSIS_BSC_ISIC4',
'STAN',
'STAN08BIS',
'STANI4',
'STANI4_2016',
'STANI4_2020',
'STANINDICATORS',
'STANINDICATORSI4',
'STAN_INDICATORS_2005',
'STAN_INDICATORS_2009',
'STAN_IO_GHG',
'STAN_IO_INTERM_M',
'STAN_IO_LEONTIEF',
'STAN_IO_LEONTIEF_DOM',
'STAN_IO_M_X',
'STAN_IO_TOT_DOM_IMP',
'STIO_2014',
'STIO_2016',
'STI_STEEL_MAKINGCAPACITY',
'STLABOUR',
'STRI',
'STRI_DIGITAL',
'STRI_H',
'STRI_H_DIGITAL',
'STRI_H_INTRAEEA',
'STRI_INTRAEEA',
'SURFACE_WATER',
'SURVEYDATA',
'TABLE1',
```

```
 'TABLE2A',
 'TABLE2B',
 'TABLE3A',
 'TABLE4',
 'TABLE5',
 'TABLE7B',
 'TABLE_I1',
 'TABLE_I2',
 'TABLE_I3',
 'TABLE_I4',
 'TABLE_I5',
 'TABLE_I6',
 'TABLE_I7',
 'TABLE_II1',
 'TABLE_II2',
 'TABLE_II3',
 'TABLE_II4',
 ...]
```

In [4]:
```
# just as a help to taking a closer look at the different data tables - so
me might have aggregated information
# or only include specific data (years/countries)
# finding groups by anticipating similar name tags
# https://stats.oecd.org/Index.aspx?DataSetCode=LAND_COVER_FUA
# https://stats.oecd.org/restsdmx/sdmx.ashx/GetDataStructure/EO43_VINTAGE
def help_analysis(keys):
    """Look at the key names and count letter combinations that appear oft
en"""
    pairs = {}
    triplets = {}
    for key in keys:
        for index, letter in enumerate(key[:-2]):
            next_letter = key[index + 1]
            pair = letter + next_letter
            next_next_letter = key[index + 2]
            triplet = pair + next_next_letter
            pairs[pair] = pairs.setdefault(pair, 0) + 1
            triplets[triplet] = triplets.setdefault(triplet, 0) + 1
    p_analysis = {k: v for k, v in sorted(pairs.items(), reverse=True, key
=lambda item: item[1])}
    trip_analysis = {k: v for k, v in sorted(triplets.items(), reverse=Tru
e, key=lambda item: item[1])}
    print(p_analysis)
    print(trip_analysis)

help_analysis(keys)
```

```
{'TA': 216, 'RE': 197, 'IN': 180, 'EO': 163, '20': 157, '_T': 150, '01': 1
40, 'A_': 133, 'LE': 131, 'EV': 129, 'AG': 127, 'AB': 127, 'BL': 127, '_I'
: 125, 'S_': 120, 'NA': 111, 'NT': 100, 'ER': 99, 'SN': 90, 'ON': 86, 'TR'
: 78, '_A': 76, 'MO': 76, 'E_': 74, '_S': 71, 'VI': 68, 'TE': 68, '_E': 64
, 'D_': 63, 'FS': 63, 'DI': 61, '_C': 61, 'RA': 61, '_V': 61, 'N2': 61, '_
2': 58, 'TI': 57, 'ND': 56, 'AT': 56, '3_': 55, 'EN': 54, '_F': 54, 'EA':
53, 'ST': 52, 'I_': 51, '_R': 51, 'IS': 50, 'AD': 49, '_P': 47, 'FI': 47,
'FF': 46, '_M': 45, 'G_': 45, 'IO': 44, 'H_': 43, 'AN': 42, 'AR': 42, 'N_'
: 41, 'MA': 40, 'NE': 39, 'R_': 37, 'IV': 37, 'T_': 37, 'EC': 37, 'AS': 37
, 'CO': 36, 'SH': 36, 'HI': 34, 'CT': 34, 'AI': 33, 'CH': 33, 'RD': 33, 'D
```

E': 33, 'SI': 32, 'SE': 32, '_D': 31, 'C_': 31, 'IC': 31, 'AL': 30, 'LA': 30, 'IT': 29, '11': 29, 'TO': 29, 'RS': 29, 'O8': 29, 'RI': 28, 'OU': 28, 'DU': 28, '12': 28, 'PA': 27, 'OR': 27, 'PE': 26, '5_': 26, '8_': 26, 'TS': 25, 'CA': 24, 'LI': 24, 'RC': 24, '1_': 24, 'IG': 24, 'PT': 24, 'ES': 24, '_G': 23, 'IL': 23, 'UR': 23, 'RN': 23, '6_': 23, 'LT': 22, '4_': 22, 'ME': 22, 'O9': 22, '_O': 21, 'FD': 21, 'V_': 21, 'GI': 21, 'E1': 21, 'AC': 21, 'L_': 21, '2_': 21, 'O1': 21, 'PO': 20, 'GE': 20, '_L': 20, 'OC': 20, 'O6': 20, 'O7': 20, 'VA': 20, 'US': 19, 'ED': 19, '7_': 19, '10': 19, 'SC': 19, 'IE': 18, 'GH': 18, 'P_': 18, 'BE': 18, 'EF': 18, '9_': 18, '0_': 18, '13': 18, 'CE': 18, 'VC': 18, 'A9': 18, 'II': 17, 'NC': 17, 'NR': 17, 'EG': 17, 'U_': 17, 'FL': 17, 'QA': 17, 'SA': 17, 'E7': 17, 'EI': 16, '_N': 16, 'EM': 16, 'BO': 16, 'VE': 16, '_B': 16, 'PR': 16, 'GL': 16, 'HA': 15, 'F_': 15, 'CI': 15, 'GR': 15, 'PP': 15, 'NV': 15, '23': 15, 'EE': 15, 'RT': 14, 'Y_': 14, 'OV': 14, 'E6': 14, 'PS': 13, 'TH': 13, '00': 13, 'K_': 13, 'UN': 13, 'SO': 13, 'M_': 13, 'HE': 12, 'SU': 12, 'GO': 12, 'NK': 12, 'OS': 11, 'VP': 11, '18': 11, 'O_': 11, 'AA': 11, 'FA': 10, 'TN': 10, 'NS': 10, 'LF': 10, '_H': 10, 'OM': 10, 'FU': 10, 'DB': 10, 'CR': 10, 'CS': 10, 'NG': 10, 'O2': 10, 'O3': 10, 'O4': 10, 'O5': 10, 'HF': 10, 'VN': 10, 'UT': 9, 'OT': 9, 'AF': 9, 'MP': 9, 'OP': 9, 'WA': 9, 'RO': 9, 'BS': 9, 'CU': 9, 'OB': 9, 'EL': 9, 'EX': 9, 'RL': 9, 'O0': 9, 'TF': 9, 'SM': 9, 'VB': 9, 'VS': 9, 'DA': 8, 'MI': 8, 'B_': 8, 'AM': 8, 'CP': 8, 'FO': 8, 'CC': 8, 'NI': 8, '15': 8, '62': 8, '72': 8, 'LO': 8, 'ID': 8, 'TY': 8, 'VM': 8, 'FE': 8, 'IA': 7, 'NU': 7, 'LC': 7, 'AP': 7, 'MT': 7, '_W': 7, 'BI': 7, 'IM': 7, 'I2': 7, 'BT': 7, 'CB': 7, '14': 7, 'SP': 7, 'UB': 7, 'ET': 7, 'TU': 7, '16': 7, '19': 7, 'OO': 7, 'EP': 7, 'TC': 7, 'OW': 7, 'AU': 7, 'OD': 7, 'VT': 7, 'E9': 7, 'SD': 6, 'AE': 6, 'IR': 6, 'BA': 6, 'S2': 6, 'DS': 6, '_U': 6, 'UL': 6, 'G2': 6, '61': 6, '71': 6, 'W_': 6, 'NO': 6, 'TP': 6, '_7': 6, 'E2': 6, 'E8': 6, 'VL': 6, 'E4': 6, '7I': 5, 'SS': 5, 'MN': 5, 'AV': 5, 'EB': 5, 'RY': 5, 'TD': 5, 'UP': 5, 'BC': 5, 'CD': 5, 'KI': 5, 'PU': 5, 'LS': 5, '25': 5, 'GB': 5, 'SV': 5, 'KE': 5, 'PD': 5, 'P2': 5, 'VG': 5, 'SL': 5, 'E3': 5, 'DD': 4, 'FR': 4, 'UM': 4, 'NB': 4, 'HR': 4, 'VD': 4, 'OF': 4, 'OA': 4, 'IX': 4, 'BU': 4, 'IF': 4, 'X_': 4, 'SK': 4, 'LL': 4, 'NN': 4, 'BN': 4, 'DN': 4, 'RF': 4, '02': 4, '80': 4, 'OK': 4, '98': 4, 'PL': 4, 'PM': 4, 'GG': 4, 'QU': 4, 'FT': 4, 'HO': 4, 'MM': 4, 'PN': 4, 'DO': 4, 'VK': 4, 'VU': 4, 'D1': 3, 'OL': 3, 'WC': 3, 'IZ': 3, 'BF': 3, 'BR': 3, 'XE': 3, 'LD': 3, 'DC': 3, 'WT': 3, 'GD': 3, '03': 3, '07': 3, '08': 3, '17': 3, '75': 3, '83': 3, '84': 3, '85': 3, '86': 3, 'TL': 3, '91': 3, '93': 3, 'XP': 3, 'TB': 3, 'JP': 3, 'KO': 3, 'LV': 3, 'NL': 3, 'NZ': 3, 'SW': 3, 'B2': 3, 'YP': 3, 'HG': 3, 'OE': 3, 'I4': 3, 'UA': 3, 'RM': 3, 'EY': 3, 'XA': 3, '2A': 3, '1C': 3, '1P': 3, '1T': 3, '2C': 3, '2P': 3, '2T': 3, 'MU': 3, '_Q': 3, 'VF': 3, 'VJ': 3, 'WS': 3, 'RG': 3, 'BD': 3, 'CX': 3, 'EU': 3, 'A2': 3, '7H': 2, '7P': 2, 'AW': 2, 'HM': 2, 'RK': 2, 'PF': 2, 'C3': 2, 'UI': 2, 'UD': 2, 'DV': 2, 'MF': 2, 'CL': 2, '04': 2, '05': 2, '06': 2, '60': 2, '63': 2, '64': 2, '65': 2, '66': 2, '67': 2, '68': 2, '69': 2, '70': 2, '73': 2, '74': 2, '76': 2, '77': 2, '78': 2, '79': 2, '81': 2, '82': 2, '88': 2, '89': 2, '90': 2, '92': 2, '95': 2, '97': 2, 'OG': 2, 'M2': 2, 'K2': 2, 'CZ': 2, 'HU': 2, '_K': 2, 'LU': 2, 'RU': 2, 'ZA': 2, 'GF': 2, 'PH': 2, 'HH': 2, 'IP': 2, 'JO': 2, 'WE': 2, 'UC': 2, '_1': 2, '1E': 2, '2E': 2, '_3': 2, '3E': 2, '_4': 2, '4E': 2, '_5': 2, '5E': 2, '_6': 2, '6E': 2, '7E': 2, '1A': 2, 'MS': 2, 'PI': 2, 'SY': 2, 'GT': 2, 'QN': 2, 'VH': 2, 'RW': 2, 'VV': 2, 'RH': 2, 'HP': 2, '4A': 2, 'E5': 2, '6A': 2, '7A': 2, '8A': 2, '9A': 2, '9B': 2, 'C1': 2, 'LH': 2, '1D': 1, 'N3': 1, 'T3': 1, 'GA': 1, 'D2': 1, 'NH': 1, 'FG': 1, 'BP': 1, 'S1': 1, 'PC': 1, 'CG': 1, 'ZE': 1, '09': 1, '21': 1, '22': 1, '24': 1, '26': 1, '27': 1, '28': 1, '29': 1, '30': 1, '31': 1, '32': 1, '33': 1, '34': 1, '35': 1, '36': 1, '37': 1, '38': 1, '39': 1, '40': 1, '41': 1, '42': 1, '43': 1, '44': 1, '45': 1, '46': 1, '47': 1, '48': 1, '49': 1, '50': 1, '51': 1, '52': 1, '53': 1, '

54': 1, '55': 1, '56': 1, '57': 1, '58': 1, '59': 1, '87': 1, 'K8': 1, '94
': 1, '96': 1, 'K9': 1, '99': 1, '_J': 1, '_Z': 1, 'FB': 1, 'AQ': 1, 'AO':
 1, 'BJ': 1, 'JE': 1, 'GS': 1, 'GV': 1, 'HC': 1, 'CQ': 1, 'WF': 1, 'FM': 1
, 'HS': 1, 'BY': 1, 'V-': 1, '-M': 1, 'BQ': 1, 'Q_': 1, 'LM': 1, 'HT': 1,
'YS': 1, 'MW': 1, 'V1': 1, 'D-': 1, '-A': 1, 'DY': 1, 'YG': 1, 'MR': 1, 'R
2': 1, 'PG': 1, 'QI': 1, 'DT': 1, 'LR': 1, 'BG': 1, 'BH': 1, 'CM': 1, 'UE'
: 1, 'FJ': 1, 'GU': 1, 'JA': 1, 'KA': 1, 'MD': 1, 'ML': 1, 'MY': 1, 'ZL':
1, 'VR': 1, 'SG': 1, 'TG': 1, 'TK': 1, 'TT': 1, 'UG': 1, 'VW': 1, 'VZ': 1,
 'RP': 1, 'SB': 1, 'N0': 1, '8B': 1, 'AK': 1, 'GC': 1, 'RV': 1, 'YD': 1, '
AX': 1, 'C2': 1, 'C4': 1, 'C5': 1, 'C6': 1, 'C7': 1, 'C8': 1, 'C9': 1, 'LY
': 1, 'TX': 1, 'XW': 1, 'WD': 1, 'WI': 1}
{'201': 137, 'REV': 131, 'TAB': 120, 'ABL': 118, 'BLE': 118, '_TA': 99, 'S
NA': 89, 'A_T': 85, 'INT': 85, '_IN': 82, 'NA_': 74, 'AGE': 73, 'MON': 62,
 '_VI': 60, 'VIN': 60, 'NTA': 60, 'ON2': 60, 'N20': 60, 'TAG': 59, '_20':
56, 'FS_': 54, 'IND': 49, 'FFS': 45, 'TRA': 44, 'RAD': 43, '_RE': 38, 'ADE
': 37, 'TER': 36, '_MA': 33, 'AG_': 33, '_TR': 33, '011': 31, 'EAG': 31, '
RD_': 30, 'EO8': 29, 'MAI': 28, 'AIN': 28, 'LE_': 28, '012': 27, '_AG': 26
, 'ERD': 25, 'ION': 23, 'NTE': 23, 'E_I': 22, 'TIO': 22, 'RS_': 22, 'ERN':
 22, 'RNE': 22, 'NET': 22, 'EO9': 22, '_AR': 21, 'IVE': 21, 'EO1': 21, '_C
O': 20, 'EO6': 20, 'EO7': 20, 'FDI': 20, 'DI_': 20, 'ARC': 19, 'RCH': 19,
'CHI': 19, 'HIV': 19, 'STA': 19, 'PER': 19, 'LE1': 19, 'S_I': 18, '13_': 1
8, 'REF': 18, 'LE7': 18, '_SN': 18, 'NA9': 18, 'A93': 18, 'NDI': 17, 'DIC'
: 17, '_EN': 17, 'ERS': 17, 'SH_': 17, 'EDU': 17, '_FL': 17, 'TIV': 17, 'Q
AS': 17, 'ASA': 17, 'SA_': 17, 'ER_': 16, 'ON_': 16, '_FI': 16, 'EVC': 16,
 'BER': 15, 'ATE': 15, 'TOR': 15, 'PAT': 15, 'ILE': 15, 'DU_': 15, '23_':
15, 'FIS': 15, 'ISH': 15, 'LE6': 15, 'TAN': 15, 'OUR': 14, 'STR': 14, 'CAT
': 14, '018': 14, '113': 14, '123': 14, 'IVA': 14, 'S_C': 13, 'TRY': 13, '
014': 13, 'FIN': 13, 'N_I': 13, 'H_A': 13, 'EAL': 13, 'IGH': 13, 'REG': 13
, 'EI_': 12, '_PA': 12, 'ICA': 12, 'S_S': 12, 'ATI': 12, '200': 12, '_ST':
 12, 'TS_': 12, 'MEI': 12, 'O10': 12, 'E_E': 12, 'ALT': 12, 'LTH': 12, 'HI
G': 12, 'GH_': 12, 'AGL': 12, 'GLI': 12, 'LIN': 12, 'INK': 12, 'NK_': 12,
'K_2': 12, 'CT_': 12, 'SCT': 12, 'TSE': 12, 'EV4': 11, '010': 11, 'ENC': 1
1, '015': 11, 'SE_': 11, '_DE': 11, 'ES_': 11, 'AD_': 11, 'S_A': 11, 'LAS'
: 11, 'ENV': 11, '18_': 11, 'ASH': 11, 'ORS': 11, 'ATO': 11, 'AT_': 11, 'G
ER': 11, 'HEA': 11, 'TH_': 11, 'TEC': 11, 'VA_': 11, 'ENT': 10, 'LFS': 10,
 'EMP': 10, '_PO': 10, 'STI': 10, '_IS': 10, '_FU': 10, '016': 10, 'ND_':
10, '_GR': 10, 'RAT': 10, 'ENR': 10, 'EO2': 10, 'EO3': 10, 'EO4': 10, 'EO5
': 10, 'FLA': 10, 'SHF': 10, 'HFI': 10, 'FIL': 10, '_EO': 10, '_PR': 10, '
NCE': 10, 'GOV': 10, 'EEN': 10, 'CE_': 10, 'IO_': 10, 'EGI': 10, 'GIO': 10
, 'A_2': 10, '_OU': 9, 'OUT': 9, 'IN_': 9, 'AN_': 9, 'D_F': 9, 'NDU': 9, '
A_S': 9, 'G_E': 9, 'NRL': 9, 'RL_': 9, '_RA': 9, 'ACT': 9, 'EO0': 9, 'I_P'
: 9, 'OV_': 9, 'ITF': 9, 'TF_': 9, '_EE': 9, 'EVB': 9, 'EVN': 9, 'EVS': 9,
 'DEE': 9, 'NV_': 9, 'V_I': 9, 'A_A': 8, 'I_I': 8, 'PAR': 8, 'ART': 8, 'NE
R': 8, 'TAL': 8, '_SU': 8, 'D_C': 8, 'ISI': 8, 'SIC': 8, 'FUN': 8, 'Y_I':
8, '_TO': 8, 'D_S': 8, 'BLI': 8, 'AND': 8, 'R_T': 8, 'SEC': 8, 'ECO': 8, '
DIO': 8, 'IOC': 8, 'OC_': 8, 'SHA': 8, 'NTR': 8, '15_': 8, '_II': 8, 'H_P'
: 8, '019': 8, 'G_2': 8, 'EVM': 8, 'ISM': 8, 'EFE': 8, 'FER': 8, 'ERE': 8,
 'REN': 8, 'E_T': 8, 'EVP': 8, 'LE4': 8, 'LE9': 8, 'TRI': 7, 'C_I': 7, 'RT
N': 7, 'TNE': 7, '_EM': 7, 'S_E': 7, 'S_P': 7, 'COM': 7, 'IC4': 7, 'UND':
7, 'DUS': 7, 'UST': 7, '_SI': 7, 'IGI': 7, 'I20': 7, 'ECT': 7, '_SE': 7, '
RE_': 7, 'G_F': 7, 'NAT': 7, 'GRA': 7, 'TR_': 7, '_AC': 7, 'SOC': 7, '16_'
: 7, 'CTR': 7, 'POS': 7, 'COL': 7, 'LAN': 7, '_DI': 7, 'CES': 7, 'EVE': 7,
 'RI_': 7, 'ITY': 7, 'NAA': 7, 'AAG': 7, 'LE8': 7, 'EVT': 7, 'WAT': 7, 'TO
U': 7, 'URI': 7, 'RIS': 7, 'SM_': 7, 'RIE': 6, 'NTS': 6, 'AFA': 6, 'FA_':
6, 'ALF': 6, 'MP_': 6, '_LA': 6, 'E_O': 6, 'TIS': 6, 'BOP': 6, 'S20': 6, '
NDS': 6, '_SO': 6, 'GID': 6, '013': 6, '017': 6, 'D_E': 6, 'CBC': 6, 'CR_'
: 6, 'CHA': 6, '_EA': 6, 'G20': 6, 'IES': 6, 'E_2': 6, 'N_S': 6, 'FIE': 6,

'IEL': 6, 'ELD': 6, 'ING': 6, '_MO': 6, 'T_O': 6, 'U_E': 6, '1_V': 6, '2_V': 6, '3_V': 6, '4_V': 6, '5_V': 6, '6_V': 6, '7_V': 6, '8_V': 6, '9_V': 6, '_LT': 6, 'I_C': 6, '_CT': 6, 'I_F': 6, 'S_B': 6, 'EST': 6, 'FSE': 6, 'T_D': 6, 'CTI': 6, 'LAC': 6, 'ERI': 6, 'CSC': 6, 'PSC': 6, 'PPP': 6, 'E62': 6, 'E72': 6, 'EVL': 6, 'LE2': 6, 'LE3': 6, '_IO': 6, '_CA': 5, 'LC_': 5, 'RIC': 5, 'RAN': 5, 'POP': 5, 'OP_': 5, 'BOU': 5, 'ITA': 5, 'IS_': 5, 'COS': 5, 'RY_': 5, 'LI2': 5, 'BTD': 5, '3_2': 5, 'BCR': 5, 'R_A': 5, 'ITI': 5, 'S_R': 5, 'CCU': 5, 'NDE': 5, 'MOB': 5, 'S_F': 5, 'L_R': 5, '_SH': 5, 'HAR': 5, 'ARE': 5, 'E_C': 5, '_NA': 5, 'URE': 5, 'TES': 5, '_PE': 5, 'G_T': 5, 'SUB': 5, 'D_A': 5, '_PU': 5, 'PUB': 5, '0_V': 5, 'EXP': 5, 'AGG': 5, 'FLO': 5, 'LOW': 5, 'OW_': 5, 'OS_': 5, 'TEM': 5, 'S_G': 5, 'RES': 5, 'ICE': 5, 'V_2': 5, 'PRO': 5, 'SIN': 5, 'VCO': 5, 'TY_': 5, 'PP2': 5, 'P20': 5, 'EFS': 5, 'SER': 5, 'TOT': 5, 'EVG': 5, 'EVI': 5, 'S_2': 5, 'ANI': 5, '6_C': 5, '8_C': 5, 'HA_': 4, '7II': 4, '3_M': 4, 'POL': 4, 'ANS': 4, 'R_E': 4, 'ONS': 4, '_GH': 4, 'GHG': 4, 'LAB': 4, 'VIT': 4, 'SUM': 4, 'AMN': 4, 'MNE': 4, 'NE_': 4, '11_': 4, 'HRS': 4, 'DUR': 4, 'EBO': 4, 'OST': 4, 'DS_': 4, 'MA_': 4, 'SIG': 4, 'TDI': 4, 'DIX': 4, 'IXE': 4, 'LEI': 4, 'III': 4, 'HAP': 4, 'APT': 4, 'PTE': 4, 'AG2': 4, '14_': 4, '_NE': 4, 'R_D': 4, '_FO': 4, 'E10': 4, 'CIT': 4, 'DAC': 4, 'N_A': 4, '_OC': 4, 'OCC': 4, 'CUP': 4, 'N_D': 4, 'C_S': 4, 'GEN': 4, 'END': 4, 'DER': 4, 'SKI': 4, 'KIL': 4, 'ILL': 4, 'LLS': 4, 'L_M': 4, 'OBI': 4, 'BIL': 4, 'E_A': 4, 'INS': 4, 'MIG': 4, 'CON': 4, 'H_D': 4, 'DEM': 4, 'UBN': 4, 'BNA': 4, 'U_G': 4, 'LTB': 4, '8_I': 4, '10_': 4, '19_': 4, 'O83': 4, 'O84': 4, 'O85': 4, 'O86': 4, 'OOK': 4, 'O91': 4, 'O93': 4, 'EPL': 4, 'PL_': 4, 'IAL': 4, 'GGR': 4, 'I_B': 4, '_BO': 4, '_EC': 4, 'SIT': 4, 'D_T': 4, 'ARG': 4, '_CH': 4, 'S_D': 4, 'S_L': 4, '_US': 4, 'QUA': 4, 'T_C': 4, 'FOR': 4, 'FTP': 4, 'TPT': 4, 'PRI': 4, 'ORD': 4, 'ABS': 4, 'IDD': 4, 'OUN': 4, 'REA': 4, '009': 4, 'USE': 4, 'INV': 4, 'AL_': 4, 'VCH': 4, '3_1': 4, '3_3': 4, '3_4': 4, '3_5': 4, '3_6': 4, '3_7': 4, 'AAN': 4, 'ANR': 4, 'NRI': 4, 'T_E': 4, 'NGL': 4, 'GLE': 4, 'OMM': 4, 'MMO': 4, 'MOD': 4, 'ODI': 4, 'DIT': 4, 'PDB': 4, 'PTR': 4, 'A_7': 4, 'E61': 4, '610': 4, '10R': 4, '620': 4, '20R': 4, '25R': 4, 'E71': 4, '710': 4, '720': 4, '_AS': 4, 'ASI': 4, 'EF_': 4, 'OTA': 4, 'EVA': 4, 'DOM': 4, 'ENU': 4, 'EVK': 4, 'LE5': 4, 'TIM': 4, '5_C': 4, '_C1': 4, '_A_': 3, 'II_': 3, 'D_D': 3, 'IEN': 3, 'CAL': 3, 'ALC': 3, 'IN3': 3, 'C_O': 3, 'UT3': 3, 'A_I': 3, 'ANN': 3, 'CAP': 3, 'GE_': 3, 'AIR': 3, 'RTR': 3, 'SSI': 3, 'P_A': 3, 'ABO': 3, 'MTA': 3, 'ANB': 3, 'NBE': 3, 'AVE': 3, 'VE_': 3, '_AN': 3, '_WA': 3, 'OMP': 3, '_EB': 3, 'OPS': 3, 'PS2': 3, 'MAR': 3, 'T_I': 3, 'D_I': 3, 'D_M': 3, 'SOF': 3, 'IZE': 3, '_CP': 3, 'CPA': 3, 'ROA': 3, 'OAD': 3, 'XE_': 3, '_I4': 3, '_ED': 3, 'UP_': 3, 'NEW': 3, 'C_E': 3, '_IT': 3, 'TIE': 3, 'CUB': 3, 'CTS': 3, 'CTO': 3, 'DEC': 3, 'TUD': 3, 'DET': 3, 'SEX': 3, 'V_D': 3, 'CD_': 3, 'NIN': 3, 'S_O': 3, '_OR': 3, 'NST': 3, '_ES': 3, 'ESO': 3, 'TUR': 3, 'N_R': 3, 'SOU': 3, 'URC': 3, 'RCE': 3, 'DEN': 3, 'G_G': 3, 'R_F': 3, 'R_R': 3, '_TS': 3, 'H_I': 3, 'U_P': 3, 'S_M': 3, 'IC_': 3, 'NVP': 3, 'TAD': 3, '01_': 3, '03_': 3, '07_': 3, '7_I': 3, '12_': 3, '17_': 3, '0_M': 3, '0_T': 3, '1_M': 3, '1_T': 3, '2_M': 3, '2_T': 3, '3_T': 3, '4_M': 3, '4_T': 3, '5_M': 3, '5_T': 3, '6_M': 3, '6_T': 3, '8_T': 3, '9_M': 3, '83_': 3, '84_': 3, '85_': 3, '86_': 3, 'O88': 3, 'O89': 3, 'O90': 3, '91_': 3, 'O92': 3, '93_': 3, 'ACC': 3, 'XP_': 3, 'CIA': 3, 'P_P': 3, 'FAC': 3, 'TBO': 3, 'HIS': 3, 'IST': 3, 'INC': 3, 'NC_': 3, '_TE': 3, 'AUS': 3, 'AUT': 3, 'CHL': 3, 'JPN': 3, 'KOR': 3, '_LV': 3, 'MEX': 3, 'S_N': 3, '_NO': 3, 'NZL': 3, 'H_F': 3, 'H_L': 3, 'REC': 3, 'STE': 3, 'ORE': 3, 'TC_': 3, 'TN_': 3, 'NAB': 3, 'BS2': 3, '007': 3, 'PRE': 3, 'RE1': 3, 'E19': 3, '198': 3, '981': 3, '_SC': 3, 'DDB': 3, 'DB2': 3, 'B20': 3, 'UNC': 3, '_TY': 3, 'TYP': 3, 'SPO': 3, 'OEC': 3, 'ECD': 3, 'I4_': 3, '4_2': 3, 'O_G': 3, 'UAL': 3, 'ORT': 3, 'M_I': 3, 'AC_': 3, 'C_R': 3, 'COV': 3, 'OVE': 3, 'VER': 3, 'TAX': 3, '11C': 3, '1CS': 3, '11P': 3, '1PS': 3, '11T': 3, '1TS': 3, '12C': 3, '2CS': 3, '12P': 3, '2PS': 3, '12T':

3, '2TS': 3, '8_R': 3, 'ATS': 3, '_QU': 3, '_7I': 3, 'I_D': 3, 'DIS': 3, '
1_A': 3, 'F_T': 3, 'VAU': 3, 'EVD': 3, 'VES': 3, 'EVF': 3, 'EVJ': 3, 'EVU'
: 3, 'SDB': 3, '_BD': 3, 'LS_': 3, 'E11': 3, 'E12': 3, 'E13': 3, 'E14': 3,
 '14A': 3, 'E6A': 3, 'E7A': 3, 'E8A': 3, 'E9A': 3, 'E9B': 3, 'OCX': 3, 'CX
_': 3, 'NI4': 3, '_DO': 3, 'I_H': 3, 'EC1': 3, 'TEN': 3, 'NUR': 3, 'TEU':
3, 'VA2': 3, 'A20': 3, '_C2': 3, '_C3': 3, 'ND1': 3, 'ULC': 3, 'A_Q': 2, '
7PS': 2, 'PSD': 2, 'AEA': 2, 'AEI': 2, 'I_N': 2, '_NU': 2, 'A_C': 2, 'MAN'
: 2, 'A_O': 2, 'AFR': 2, 'LIS': 2, 'IRT': 2, 'IR_': 2, 'MIS': 2, 'SIO': 2,
 'UMT': 2, 'AB_': 2, 'B_A': 2, 'N_P': 2, '1_R': 2, 'D_R': 2, '_DU': 2, 'E_
H': 2, 'AWC': 2, 'WCO': 2, 'BAT': 2, 'ARK': 2, 'ST_': 2, 'TOE': 2, 'SIZ':
2, 'F_S': 2, '_I3': 2, 'C3_': 2, 'ED_': 2, 'D_2': 2, 'BUI': 2, 'UIL': 2, '
ILT': 2, 'LT_': 2, 'T_U': 2, '_UP': 2, 'P_F': 2, 'FUA': 2, 'ULA': 2, 'TED'
: 2, 'EII': 2, 'LEV': 2, 'R_B': 2, 'R_C': 2, 'FOB': 2, 'TIC': 2, 'PSE': 2,
 'CRS': 2, 'RS1': 2, 'GRE': 2, 'REQ': 2, 'CSE': 2, '_CI': 2, 'ETR': 2, 'AC
I': 2, 'EC_': 2, 'C_C': 2, 'EN_': 2, 'C_D': 2, 'LD_': 2, 'STU': 2, 'UPA':
2, 'X_A': 2, 'UR_': 2, 'DV_': 2, '_DC': 2, 'DCD': 2, '_GE': 2, 'D_P': 2, '
_PP': 2, '_AL': 2, 'EAR': 2, 'LES': 2, 'L_S': 2, 'N_N': 2, 'R_S': 2, 'G_M'
: 2, '_MI': 2, 'IGR': 2, 'G_N': 2, 'EAC': 2, 'G_P': 2, 'WT_': 2, 'ORG': 2,
 'MFP': 2, '_ME': 2, 'NSH': 2, '_CL': 2, 'ASS': 2, 'L_A': 2, 'EGD': 2, 'GD
N': 2, 'DNA': 2, 'A_P': 2, 'UBL': 2, 'LIC': 2, 'OCI': 2, 'SPE': 2, 'NSI':
2, 'VPE': 2, 'RFI': 2, '008': 2, '02_': 2, '04_': 2, '05_': 2, '06_': 2, '
08_': 2, '0_I': 2, '1_I': 2, '2_I': 2, '103': 2, '3_I': 2, '3_L': 2, '4_I'
: 2, '5_I': 2, '6_I': 2, '107': 2, 'ET_': 2, '20_': 2, 'O60': 2, '60_': 2,
 'O61': 2, '61_': 2, 'O62': 2, '62_': 2, 'O63': 2, '63_': 2, 'O64': 2, '64
_': 2, 'O65': 2, '65_': 2, 'O66': 2, '66_': 2, 'O67': 2, '67_': 2, '7_M':
2, '7_T': 2, 'O68': 2, '68_': 2, '8_M': 2, 'O69': 2, '69_': 2, '9_T': 2, '
O70': 2, '70_': 2, 'O71': 2, '71_': 2, 'O72': 2, '72_': 2, 'O73': 2, '73_'
: 2, 'O74': 2, '74_': 2, 'O75': 2, '75_': 2, 'O76': 2, '76_': 2, 'O77': 2,
 '77_': 2, 'O78': 2, '78_': 2, 'O79': 2, '79_': 2, 'O80': 2, '80_': 2, 'O8
1': 2, '81_': 2, 'O82': 2, '82_': 2, '3_F': 2, '7_O': 2, 'UTL': 2, 'TLO':
2, 'LOO': 2, '88_': 2, '89_': 2, '9_I': 2, '90_': 2, '92_': 2, 'O95': 2, '
95_': 2, 'O97': 2, '97_': 2, 'EPE': 2, 'L_O': 2, 'ERT': 2, 'TCR': 2, 'G_S'
: 2, 'CTB': 2, 'BOO': 2, 'OK2': 2, 'K20': 2, 'I_A': 2, 'UMM': 2, 'P_I': 2,
 '_HI': 2, 'W_C': 2, 'W_I': 2, 'C_A': 2, 'OSI': 2, '_AU': 2, 'BEL': 2, 'BR
A': 2, 'CAN': 2, 'CHE': 2, 'CHN': 2, 'CZE': 2, 'DEU': 2, 'DNK': 2, 'ESP':
2, '_FR': 2, 'FRA': 2, '_GB': 2, 'GBR': 2, 'GRC': 2, 'HUN': 2, 'IDN': 2, '
OR_': 2, 'ETA': 2, 'TAI': 2, 'AIL': 2, 'IRL': 2, 'ISL': 2, 'ISR': 2, 'LTU'
: 2, 'LUX': 2, 'LVA': 2, 'NLD': 2, 'NOR': 2, 'PRT': 2, '_SV': 2, 'SVK': 2,
 'SVN': 2, 'SWE': 2, 'S_T': 2, 'USA': 2, 'ZAF': 2, 'H_E': 2, 'MPL': 2, '_F
S': 2, 'OL_': 2, 'DEV': 2, 'DIF': 2, 'IFF': 2, 'T_R': 2, 'H_R': 2, 'OOD':
2, 'AST': 2, 'PTC': 2, 'PTN': 2, 'GBA': 2, 'D_N': 2, 'BAR': 2, 'ARD': 2, '
SCI': 2, 'CIE': 2, 'GRR': 2, 'SSE': 2, 'VC_': 2, 'NCT': 2, '_AM': 2, 'AMO'
: 2, 'MOU': 2, 'UNT': 2, 'YPE': 2, 'PE_': 2, 'COR': 2, 'VNG': 2, 'ROT': 2,
 '_DA': 2, '202': 2, '020': 2, 'HOU': 2, 'ICT': 2, '_BU': 2, 'BUS': 2, 'VP
T': 2, 'PT_': 2, 'IOT': 2, 'OTS': 2, 'SI4': 2, 'HG_': 2, 'RTA': 2, 'CAS':
2, 'NSP': 2, 'POR': 2, 'F_I': 2, 'DAT': 2, 'ATA': 2, 'SEN': 2, 'NGE': 2, '
F_R': 2, '_RO': 2, 'ERM': 2, 'RM_': 2, 'JOB': 2, 'OBQ': 2, 'KEY': 2, 'EY_'
: 2, 'EG_': 2, 'VME': 2, 'EXA': 2, 'XAG': 2, '_I_': 2, 'I_R': 2, 'MAT': 2,
 'I_T': 2, 'MEM': 2, 'IG_': 2, 'P_E': 2, 'DUC': 2, 'UCA': 2, '_1E': 2, '1E
E': 2, '_2E': 2, '2EE': 2, '_3E': 2, '3EE': 2, '_4E': 2, '4EE': 2, '_5E':
2, '5EE': 2, '_6E': 2, '6EE': 2, '_7E': 2, '7EE': 2, '11A': 2, '1AA': 2, '
I_E': 2, '12A': 2, '2AA': 2, '3_R': 2, '4_R': 2, '5_R': 2, '6_R': 2, '7_R'
: 2, '7_S': 2, '8_S': 2, '9_R': 2, '9_S': 2, 'TI_': 2, 'PI_': 2, 'NOV': 2,
 'ONR': 2, 'NRD': 2, 'DBI': 2, 'DB_': 2, 'S_Q': 2, 'ALI': 2, 'LIF': 2, 'PM
R': 2, 'PNN': 2, '005': 2, 'TRC': 2, 'RCC': 2, 'ISC': 2, '625': 2, '725':
2, 'E80': 2, '801': 2, 'QNA': 2, 'IVI': 2, 'EMO': 2, 'MOG': 2, 'N_E': 2, '
_AP': 2, 'VEN': 2, 'EVH': 2, 'VIS': 2, 'VJP': 2, 'SIA': 2, 'IAN': 2, 'VKO'

: 2, 'VNZ': 2, 'VPO': 2, 'VPR': 2, 'VSL': 2, 'VSV': 2, 'VSW': 2, 'VTU': 2,
 'EVV': 2, 'E_F': 2, 'RHP': 2, 'HPI': 2, 'RGE': 2, 'TA_': 2, 'DBS': 2, 'BS
_': 2, 'BDI': 2, 'TRE': 2, 'A_F': 2, 'GI2': 2, 'MES': 2, '1_S': 2, '2_A':
2, '2_S': 2, '3_A': 2, '3_S': 2, '4A_': 2, 'E1_': 2, 'E2_': 2, 'E3_': 2, '
E5_': 2, '6A_': 2, '7A_': 2, '8A_': 2, '9A_': 2, '9B_': 2, 'OCR': 2, 'SIS'
: 2, '_BS': 2, 'BSC': 2, 'O_L': 2, '_LE': 2, 'LEO': 2, 'EON': 2, 'ONT': 2,
 'NTI': 2, 'IEF': 2, 'O_2': 2, 'DIG': 2, 'GIT': 2, '_H_': 2, 'RAE': 2, 'AE
E': 2, 'EEA': 2, 'SUR': 2, 'II1': 2, 'II2': 2, 'II3': 2, 'EC2': 2, 'EC4':
2, 'EUS': 2, 'USD': 2, 'IME': 2, 'ISP': 2, '_C4': 2, '_C5': 2, 'USL': 2, '
SLH': 2, 'LHR': 2, '7HA': 1, '7IA': 1, 'IA_': 1, 'SD_': 1, '_D1': 1, 'D1D'
: 1, '1D4': 1, 'NUT': 1, 'UTR': 1, 'I_O': 1, '_OT': 1, 'OTH': 1, 'THE': 1,
 'HER': 1, 'N3_': 1, 'ANU': 1, 'NUF': 1, 'T3_': 1, '3_P': 1, 'AFD': 1, 'FD
D': 1, 'DDA': 1, 'DAN': 1, 'FRI': 1, 'APO': 1, 'OLI': 1, 'E_G': 1, '_GA':
1, 'GAP': 1, 'NS_': 1, 'CO2': 1, 'EMI': 1, 'ISS': 1, 'R_G': 1, 'P_L': 1, '
P_V': 1, 'UT_': 1, 'T_P': 1, 'RD2': 1, 'D20': 1, 'EV3': 1, 'EV2': 1, 'ANH'
: 1, 'NHR': 1, 'AVD': 1, 'VD_': 1, '_HR': 1, 'AV_': 1, 'V_A': 1, 'N_W': 1,
 'WAG': 1, 'COU': 1, '002': 1, 'BEN': 1, 'NCH': 1, 'CHM': 1, 'HMA': 1, 'RK
_': 1, 'K_S': 1, 'OF_': 1, 'BFG': 1, 'FGI': 1, 'BFS': 1, 'FSI': 1, 'BIM':
1, 'IMT': 1, 'MTS': 1, 'BPF': 1, 'PF1': 1, 'BRO': 1, 'ADB': 1, 'DBA': 1, '
BAN': 1, '_DB': 1, 'BSI': 1, 'IC3': 1, 'TD_': 1, 'LCU': 1, 'CUL': 1, 'LAT'
: 1, 'EIV': 1, 'A_E': 1, '4_N': 1, '_B_': 1, 'B_E': 1, '_C_': 1, '_D_': 1,
 'CIF': 1, 'IF_': 1, 'F_F': 1, 'OB_': 1, 'B_I': 1, 'CII': 1, 'IIE': 1, 'IE
1': 1, 'CPL': 1, 'CPS': 1, 'S1_': 1, '1_G': 1, 'CSP': 1, 'SPC': 1, 'PCU':
1, 'UBE': 1, '_ET': 1, 'ACD': 1, 'CDE': 1, 'DEF': 1, 'EFL': 1, 'ACG': 1, '
CGE': 1, 'GEO': 1, 'CIN': 1, 'ACS': 1, 'TIZ': 1, 'ZEN': 1, 'URA': 1, 'TAY'
: 1, 'C_F': 1, 'UDY': 1, 'C_L': 1, '_LF': 1, 'EX_': 1, 'R_I': 1, 'D_G': 1,
 'PPF': 1, 'PFD': 1, 'G_A': 1, 'ARN': 1, 'RNI': 1, 'NGS': 1, 'EA_': 1, '_S
K': 1, 'LDS': 1, 'ORI': 1, 'RIG': 1, 'GIN': 1, 'TE_': 1, 'TEG': 1, 'EGO':
1, 'GOR': 1, 'ORY': 1, 'NNE': 1, 'NEX': 1, 'EX2': 1, 'ATU': 1, 'UDE': 1, '
G_I': 1, 'IT_': 1, 'T_A': 1, 'ALL': 1, 'NEA': 1, 'G_W': 1, '_WT': 1, 'EAM'
: 1, 'AMF': 1, 'AR_': 1, 'R_M': 1, 'U_C': 1, 'CLA': 1, 'U_D': 1, 'L_F': 1,
 'L_I': 1, 'U_F': 1, 'ANA': 1, 'CIO': 1, 'ELS': 1, 'LSP': 1, 'PEN': 1, 'EN
S': 1, 'ERF': 1, 'C_T': 1, 'O01': 1, 'O02': 1, 'O03': 1, 'O04': 1, 'O05':
1, 'O06': 1, 'O07': 1, 'O08': 1, 'O09': 1, '09_': 1, '100': 1, '00_': 1, '
101': 1, '102': 1, '104': 1, '105': 1, '106': 1, 'T_1': 1, 'T_2': 1, '108'
: 1, 'O11': 1, 'O12': 1, 'O13': 1, 'O14': 1, 'O15': 1, 'O16': 1, 'O17': 1,
 'O18': 1, 'O19': 1, 'O20': 1, 'O21': 1, '21_': 1, 'O22': 1, '22_': 1, 'O2
3': 1, 'O24': 1, '24_': 1, 'O25': 1, '25_': 1, 'O26': 1, '26_': 1, 'O27':
1, '27_': 1, 'O28': 1, '28_': 1, 'O29': 1, '29_': 1, 'O30': 1, '30_': 1, '
O31': 1, '31_': 1, 'O32': 1, '32_': 1, 'O33': 1, '33_': 1, 'O34': 1, '34_'
: 1, 'O35': 1, '35_': 1, 'O36': 1, '36_': 1, 'O37': 1, '37_': 1, 'O38': 1,
 '38_': 1, 'O39': 1, '39_': 1, 'O40': 1, '40_': 1, 'O41': 1, '41_': 1, 'O4
2': 1, '42_': 1, 'O43': 1, '43_': 1, 'O44': 1, '44_': 1, 'O45': 1, '45_':
1, 'O46': 1, '46_': 1, 'O47': 1, '47_': 1, 'O48': 1, '48_': 1, 'O49': 1, '
49_': 1, 'O50': 1, '50_': 1, 'O51': 1, '51_': 1, 'O52': 1, '52_': 1, 'O53'
: 1, '53_': 1, 'O54': 1, '54_': 1, 'O55': 1, '55_': 1, 'O56': 1, '56_': 1,
 'O57': 1, '57_': 1, 'O58': 1, '58_': 1, 'O59': 1, '59_': 1, '4_F': 1, '5_
F': 1, '6_F': 1, 'O87': 1, '87_': 1, 'OK8': 1, 'K87': 1, '8_F': 1, '9_F':
1, '0_F': 1, '1_F': 1, '1_L': 1, '2_F': 1, 'O94': 1, '94_': 1, '5_L': 1, '
O96': 1, '96_': 1, 'OK9': 1, 'K97': 1, 'O98': 1, '98_': 1, 'O99': 1, '99_'
: 1, 'PEA': 1, 'L_C': 1, '_CD': 1, '_OV': 1, 'L_T': 1, 'EPS': 1, 'ETC': 1,
 'P_C': 1, 'COF': 1, 'OFO': 1, 'FOG': 1, 'OG_': 1, '_SP': 1, 'PEC': 1, 'EC
I': 1, 'P_M': 1, 'MOR': 1, 'RSC': 1, '_PM': 1, 'PM2': 1, 'M2_': 1, '2_5':
1, '4_P': 1, '5_P': 1, 'FAM': 1, 'AMI': 1, 'MIL': 1, 'ILY': 1, 'DII': 1, '
IIN': 1, 'DEX': 1, 'GR_': 1, 'IIP': 1, 'Y_E': 1, 'CO_': 1, 'O_H': 1, 'W_A'
: 1, 'W_P': 1, '_BE': 1, '_BR': 1, '_CZ': 1, '_DN': 1, 'S_H': 1, '_HU': 1,
 '_ID': 1, 'LED': 1, '_IR': 1, 'S_J': 1, '_JP': 1, 'S_K': 1, '_KO': 1, '_L

U': 1, '_NL': 1, '_NZ': 1, '_RU': 1, 'RUS': 1, '_SW': 1, '_TU': 1, 'S_U':
1, 'S_Z': 1, '_ZA': 1, '_FA': 1, '_FB': 1, 'FBS': 1, '_AQ': 1, 'AQU': 1, '
FLD': 1, 'FLE': 1, 'LEE': 1, 'EET': 1, 'H_G': 1, '_GF': 1, 'GFT': 1, 'INL'
: 1, 'NLA': 1, '_RD': 1, 'ECR': 1, 'CRE': 1, 'H_T': 1, 'OBS': 1, 'FOO': 1,
 'OD_': 1, 'D_W': 1, 'WAS': 1, 'FSS': 1, '0_P': 1, 'BAO': 1, 'AOR': 1, 'NT
1': 1, 'D_O': 1, '_OB': 1, 'OBJ': 1, 'BJE': 1, 'JEC': 1, 'E_N': 1, 'SEO':
1, 'GFG': 1, 'ID2': 1, 'DEB': 1, 'EBT': 1, 'V_L': 1, 'V_S': 1, 'SEA': 1, '
V_W': 1, '_WB': 1, 'REE': 1, 'N_G': 1, 'GRO': 1, 'ROW': 1, 'OWT': 1, 'WTH'
: 1, 'GSS': 1, 'GVC': 1, 'E_R': 1, 'EMR': 1, 'H_H': 1, '_HC': 1, 'HCQ': 1,
 'CQI': 1, 'LTC': 1, 'LVN': 1, '_PH': 1, 'PHM': 1, 'HMC': 1, 'ROC': 1, 'H_
S': 1, 'TAT': 1, 'H_W': 1, '_WF': 1, 'WFM': 1, 'FMI': 1, 'HGR': 1, 'HH_':
1, 'DAS': 1, 'STP': 1, 'TPO': 1, 'URS': 1, 'RSP': 1, 'POV': 1, 'OUS': 1, '
E_P': 1, 'HSL': 1, 'T_B': 1, 'T_H': 1, '_HH': 1, 'HH2': 1, 'IMW': 1, '_OE'
: 1, 'TAC': 1, 'T_F': 1, 'T_L': 1, 'DSC': 1, 'SCA': 1, 'APE': 1, 'PES': 1,
 'TSI': 1, 'IPM': 1, 'PM_': 1, 'M_S': 1, 'ASU': 1, 'SUA': 1, 'L_B': 1, '_B
Y': 1, 'BY_': 1, 'Y_A': 1, 'F_A': 1, 'CCE': 1, 'ESS': 1, 'F_C': 1, '_CQ':
1, 'F_G': 1, '_GO': 1, 'GOO': 1, 'ODS': 1, 'NV-': 1, 'V-M': 1, '-MT': 1, '
MTN': 1, 'F_P': 1, 'PAS': 1, 'ENG': 1, 'CCI': 1, 'CID': 1, 'IDE': 1, 'D_H'
: 1, '_HA': 1, 'HAU': 1, 'AUL': 1, 'LAG': 1, 'SHO': 1, 'HOR': 1, 'RT_': 1,
 'T_T': 1, 'BQ_': 1, 'Q_I': 1, 'KEI': 1, 'Y_S': 1, 'B_R': 1, 'G_V': 1, '_V
A': 1, 'VAC': 1, 'HAN': 1, 'ANG': 1, 'D_U': 1, 'VEL': 1, 'LMP': 1, 'MPE':
1, 'PEX': 1, 'RIA': 1, 'A_D': 1, 'OP6': 1, '_BT': 1, 'BTS': 1, 'CLI': 1, '
Y_W': 1, '_WE': 1, 'WEI': 1, 'EIG': 1, 'GHT': 1, 'HTS': 1, 'PPI': 1, 'TRD'
: 1, 'EM4': 1, 'EMT': 1, 'MET': 1, 'NUP': 1, 'P_R': 1, 'MIN': 1, 'IN2': 1,
 'N2A': 1, '2AV': 1, 'SMA': 1, 'ATC': 1, 'TCH': 1, 'MST': 1, 'MTC': 1, 'MT
P': 1, 'TPI': 1, 'MUL': 1, 'ULT': 1, 'LTI': 1, 'ISY': 1, 'SYS': 1, 'YST':
1, 'MUN': 1, 'UNW': 1, 'MW_': 1, '_CU': 1, 'CUR': 1, 'URP': 1, '5_N': 1, '
OV1': 1, 'V15': 1, 'NCC': 1, 'NRR': 1, 'OCE': 1, 'CEA': 1, 'EAN': 1, 'CD-'
: 1, 'D-A': 1, '-AE': 1, 'SE2': 1, 'E20': 1, 'PAG': 1, 'RT2': 1, 'COO': 1,
 'OOP': 1, '_IP': 1, 'IPC': 1, 'BI_': 1, 'B_G': 1, 'B_L': 1, 'PDY': 1, 'DY
G': 1, 'YGT': 1, 'GTH': 1, 'MR2': 1, 'R20': 1, 'NNI': 1, 'NI_': 1, 'NN_':
1, 'FIV': 1, 'ROJ': 1, 'PPG': 1, 'PGD': 1, 'GDP': 1, 'PPR': 1, 'PRF': 1, '
CPI': 1, 'ROF': 1, 'OFS': 1, 'FSV': 1, 'SVC': 1, 'OTE': 1, 'CTE': 1, 'EAS'
: 1, 'PT1': 1, 'PT2': 1, 'PT3': 1, 'PT4': 1, 'PT5': 1, 'PT6': 1, 'PT7': 1,
 'PT8': 1, 'PT9': 1, 'CCS': 1, 'CSA': 1, 'TRS': 1, 'RSA': 1, 'TRU': 1, 'RU
B': 1, '_7H': 1, '7HH': 1, 'SCR': 1, 'E7P': 1, 'QIT': 1, 'ITS': 1, 'RDS':
1, 'DSU': 1, 'RDT': 1, 'DTA': 1, 'Y_P': 1, '_GL': 1, '_MS': 1, 'MSI': 1, '
ALO': 1, 'LOF': 1, 'OFF': 1, 'FFI': 1, 'FIC': 1, 'ICI': 1, 'ALR': 1, 'LRE'
: 1, 'ECP': 1, 'CPT': 1, 'PTS': 1, '_OD': 1, 'ODF': 1, 'OGR': 1, 'ONO': 1,
 'NOM': 1, 'INN': 1, 'NNO': 1, 'OVA': 1, 'VAT': 1, 'N_L': 1, 'N_M': 1, 'AN
T': 1, 'N_T': 1, 'YPO': 1, 'G_B': 1, 'USI': 1, 'SI_': 1, 'RET': 1, 'VAR':
1, 'US_': 1, 'VBA': 1, 'VBE': 1, 'VBF': 1, 'BFA': 1, 'VBG': 1, 'BGR': 1, '
VBH': 1, 'BHS': 1, 'VBL': 1, 'BLZ': 1, 'VBO': 1, 'BOL': 1, 'VBR': 1, 'VBT'
: 1, 'BTN': 1, 'VCA': 1, 'VCI': 1, 'CIV': 1, 'VCM': 1, 'CMR': 1, 'COD': 1,
 'COG': 1, 'COK': 1, 'VCP': 1, 'CPV': 1, 'VCR': 1, 'CRI': 1, 'VCU': 1, 'VC
Z': 1, 'VDE': 1, 'VDN': 1, 'VDO': 1, 'VEC': 1, 'ECU': 1, 'VEG': 1, 'EGY':
1, 'NUE': 1, 'UE_': 1, 'VEU': 1, 'VFI': 1, 'VFJ': 1, 'FJI': 1, 'VFR': 1, '
VGB': 1, 'VGH': 1, 'GHA': 1, 'VGR': 1, 'VGT': 1, 'GTM': 1, 'VGU': 1, 'GUY'
: 1, 'VHO': 1, 'HON': 1, 'VHU': 1, 'VID': 1, 'VIR': 1, 'VJA': 1, 'JAM': 1,
 'PN_': 1, 'VKA': 1, 'KAZ': 1, 'VKE': 1, 'KEN': 1, 'VLC': 1, 'LCA': 1, 'VL
I': 1, 'LIE': 1, 'VLS': 1, 'LSO': 1, 'VLT': 1, 'VLU': 1, 'VLV': 1, 'VMA':
1, 'VMD': 1, 'MDG': 1, 'VML': 1, 'MLI': 1, 'VMN': 1, 'MNG': 1, 'VMU': 1, '
MUS': 1, 'VMY': 1, 'MYS': 1, 'VNA': 1, 'NAM': 1, 'VNE': 1, 'NGA': 1, 'VNI'
: 1, 'NIC': 1, 'VNL': 1, 'VNO': 1, 'VNR': 1, 'NRU': 1, 'ZL_': 1, 'VPA': 1,
 'PAN': 1, 'VPH': 1, 'PHL': 1, 'VPN': 1, 'PNG': 1, 'PRY': 1, 'EVR': 1, 'VR
W': 1, 'RWA': 1, 'VSE': 1, 'VSG': 1, 'SGP': 1, 'SLB': 1, 'SLV': 1, 'SWZ':
1, 'VSY': 1, 'SYC': 1, 'VTC': 1, 'TCD': 1, 'VTG': 1, 'TGO': 1, 'VTH': 1, '

```
THA': 1, 'VTK': 1, 'TKL': 1, 'VTT': 1, 'TTO': 1, 'TUN': 1, 'VUG': 1, 'UGA'
: 1, 'VUR': 1, 'URY': 1, 'VUS': 1, 'VVE': 1, 'VVU': 1, 'VUT': 1, 'EVW': 1,
 'VWS': 1, 'WSM': 1, 'EVZ': 1, 'VZA': 1, 'FIT': 1, 'RFD': 1, 'IN1': 1, 'TA
R': 1, 'GET': 1, 'RIO': 1, 'IOM': 1, 'OMA': 1, 'RKE': 1, 'KER': 1, 'RMW':
1, 'RPE': 1, 'RSL': 1, 'SLA': 1, '_AF': 1, 'GBL': 1, 'RWB': 1, 'SBE': 1, '
EG2': 1, '_FP': 1, 'A_H': 1, '_HK': 1, 'ITC': 1, 'SME': 1, 'SCO': 1, 'REB'
: 1, 'BOA': 1, 'OAR': 1, '0_A': 1, '0_S': 1, 'E29': 1, 'E30': 1, 'E31': 1,
 'E40': 1, 'E41': 1, 'E42': 1, 'E43': 1, 'E44': 1, 'E45': 1, '5_A': 1, '5_
S': 1, 'E6_': 1, '6_S': 1, 'E75': 1, '750': 1, 'E7_': 1, 'E8_': 1, 'B_S':
1, 'E9_': 1, 'SNG': 1, 'NGF': 1, 'X_D': 1, 'X_R': 1, 'SC_': 1, 'AN0': 1, '
N08': 1, '08B': 1, '8BI': 1, 'BIS': 1, 'RSI': 1, 'O_I': 1, 'M_M': 1, 'F_D'
: 1, 'O_M': 1, '_M_': 1, 'M_X': 1, 'O_T': 1, 'OT_': 1, 'OM_': 1, '_IM': 1,
 'IMP': 1, 'I_S': 1, 'TEE': 1, 'EEL': 1, 'EL_': 1, 'MAK': 1, 'AKI': 1, 'KI
N': 1, 'NGC': 1, 'GCA': 1, 'APA': 1, 'PAC': 1, 'STL': 1, 'TLA': 1, 'URF':
1, 'RFA': 1, 'ACE': 1, 'E_W': 1, 'URV': 1, 'RVE': 1, 'VEY': 1, 'EYD': 1, '
YDA': 1, 'E2A': 1, 'E2B': 1, 'E3A': 1, 'E7B': 1, '_I1': 1, '_I2': 1, '_I5'
: 1, '_I6': 1, '_I7': 1, 'II4': 1, 'NVI': 1, 'C_2': 1, 'AXA': 1, 'XAU': 1,
 'UTO': 1, 'C10': 1, '0_R': 1, 'C1_': 1, 'C2_': 1, '2_R': 1, 'EC3': 1, 'C4
_': 1, 'EC5': 1, 'C5_': 1, 'EC6': 1, 'C6_': 1, 'EC7': 1, 'C7_': 1, 'EC8':
1, 'C8_': 1, 'EC9': 1, 'C9_': 1, 'P_D': 1, '_AV': 1, 'E_D': 1, 'FRE': 1, '
ES3': 1, 'EUR': 1, 'POT': 1, 'SDC': 1, 'DCB': 1, 'CBR': 1, 'SDE': 1, 'IM2'
: 1, 'M20': 1, 'MEL': 1, 'ELY': 1, 'LY_': 1, 'Y_B': 1, 'BDS': 1, 'ME_': 1,
 'E_U': 1, 'IM_': 1, 'M_2': 1, 'SP_': 1, 'ISX': 1, 'A_N': 1, 'NOW': 1, 'OW
C': 1, 'WCA': 1, 'M_D': 1, 'OME': 1, 'M_E': 1, 'INB': 1, 'NBO': 1, 'M_K':
1, '_KE': 1, '_PC': 1, 'M_N': 1, 'ONA': 1, 'NAL': 1, 'L_E': 1, 'M_O': 1, '
UTB': 1, 'M_R': 1, '_EX': 1, 'D10': 1, 'D12': 1, 'ND2': 1, 'ND3': 1, 'ND4'
: 1, 'ND6': 1, 'ND8': 1, 'ND9': 1, 'T_S': 1, 'TXW': 1, 'XWD': 1, 'WDE': 1,
 'EEQ': 1, 'C_Q': 1, '_GI': 1, 'U_S': 1, 'NVE': 1, '_AB': 1, 'BST': 1, 'RA
C': 1, 'BOD': 1, 'SCH': 1, 'EAT': 1, 'R_U': 1, 'WEA': 1, 'WIL': 1, 'ILD':
1, 'D_L': 1, '_LI': 1, 'IFE': 1, 'WSD': 1, 'WSE': 1, 'RG_': 1}
```

Auf Basis dieser kleinen Auswertung und dem nachgucken der einiger Keys auf der Seite der OECD, haben wir einige keys "ausgeschlossen". Primär, weil die Daten gedoppelt vorkamen. Die Funktionen zum aussortieren und herunterladen haben wir zu Gunsten der Übersichtlichkeit in einem separaten Dokument geschrieben. Zu finden sind diese in "function_downloader.py".

Anschließend haben wir den Downloader als Skript geschrieben und initiiert: "Downloader_Script.py" Dieses Skript konnten wir dann auf dem Server und all unseren unseren Rechnern ausführen, sodass überall ein gleichnamiger Ordner mit den Datein liegt um weiter damit arbeiten zu können.

Für eine Übersicht über die uns zur Verfügung stehenden Daten bzw. den Spaltennamen haben wir folgende kleine Analyse verwendet:

In [15]:
```
from Global_configuartions import FOLDERNAME_RAW, FOLDERNAME_PREP
from pathlib import Path
from collections import Counter
import pandas as pd

def analyse_columns(keys, foldername):
    """identify all remaining column names of all files"""
    col_names = []
    path = Path(foldername)
    pbar = tqdm(keys)
```

```
    for key in pbar:
        pbar.set_description(key)
        file_key = path / f"{key}.csv"
        df = pd.read_csv(file_key, nrows=1)
        key_col_names = df.columns.to_list()
        col_names.extend(key_col_names)

    col_count = Counter(col_names)
    return pd.DataFrame.from_dict(col_count, orient="index").sort_values(b
y=0, ascending=False)

path_raw = Path(FOLDERNAME_RAW)
path_prep = Path(FOLDERNAME_PREP)

# list of all keys we have downloaded
keys_downloaded = [x.name.rstrip(".csv") for x in path_raw.iterdir() if x.
is_file()]

# run analysis on keys to find out frequency of different column-names
column_overview = analyse_columns(keys_downloaded, FOLDERNAME_RAW)
column_overview
```

WSECTOR: 100%|██████████| 529/529 [00:05<00:00, 92.83it/s]

Out[15]:

	0
Flag Codes	528
Value	528
Flags	528
Country	402
Unit	359
...	...
Injury type	1
INJURY_TYPE	1
To: (sector in column)	1
COL	1
ISIC	1

713 rows × 1 columns

Mithilfe der Funktion sowie der regelmäßigen Betrachtung der Daten haben wir iterativ Funktionen geschrieben, um die Daten aufzubereiten. Die Funktionen finden sich inhaltlich getrennt in vier Bereiche in den folgenden Dateien:

1. Allgemeines "ausmisten" der Spalten in "functions_data_prep_overall.py"
2. Identifizieren und Benennen der Spalte die Informationen über die Jahre beinhaltet: "functions_data_prep_annual_data.py"
3. Identifizieren, relevante Einträge aufwählen und Benennen der Spalte die Informationen über

das Land beinhaltet: "functions_data_prep_country_data.py"
4. Datensätze so vorbereiten, dass die Feature-Spalten benannt sind und die Einträge aggregiert sind. "functions_data_prep_feature_shape.py"

Das Skript "Preparation-Script.py" nutzt diese Funktionen, bereitet die jeweiligen Datensätze auf und speichert sie in einem Ordner. In diesen Ordner legen wir auch die aufbereiteten Weltbankdaten und können mit der Modellierung beginnen. Es resultiert eine csv Datei die zu jedem Dataset den Status der transformation beinhaltet.

Die Datei "Pisa_exploration.ipynb" beinhaltet unsere Analyse der Pisa Daten.

Modellierung

Input Daten - Erstellung der verschiedenen Input Daten die wir für den Vergleich der Modelle nutzen möchten jeweils asl csv.

Zur Ausführbarkeit auf dem Server sowie unserer aller Rechner wurde hier natürlich wieder ein Skript verwendet. Die Funktionen hierfür sind in "functions_input_data.py" gespeichert. Der Inhalt des Skript namens "Input_Data-Script.py" sowie einige Betrachtungen zu den Ergebnissen werden hier ausgeführt.

In [14]:
```
from functions_input_data import *
from pathlib import Path
import itertools
from tqdm import tqdm

# Definition of several parameters
Input_Data = ["aggregated-years", "pisa-years"]
Pisa_Type = ["MATH", "READ", "SCIENCE"]
path_0 = Path(FOLDERNAME_MODELLING_DATA)
path_input = Path(path_0/"Input Data")

# location of pisa data
path_pisa = Path(FOLDERNAME_PISA)
path_final_data = Path(FOLDERNAME_FINAL_DATA)

# percentages of non-na Data we want to compare
Iterations = np.arange(50, 105, 5)

# remove for script - overview table of feature information
index = itertools.product(Input_Data, Pisa_Type, Iterations)
overview_input_data = pd.DataFrame(columns=["Inpu_Data_Type", "Pisa-Type",
 "Percentage_non_NA", "Number_Features",
                                             "Average_Percentage_NAs", "Ave
rage_Percentage_apprximated_Interpolation",
                                             "Average_Percentage_approximat
ed_Annual_Mean"], index=pd.MultiIndex.from_tuples(index))

# For each type of input Data
InputData = tqdm(Input_Data)
for input_data in InputData:
    InputData.set_description(f"{input_data}")

    # create Folders - one Folder per Input type
```

```
    path_0.mkdir(exist_ok=True)
    path_input.mkdir(exist_ok=True)
    path_data_type = Path(path_input / input_data)
    path_data_type.mkdir(exist_ok=True)

    # for every Pisa type seperately, as they start in different years
    pbar0 = tqdm(Pisa_Type)
    for pisa_type in pbar0:
        pbar0.set_description(f"{input_data}-{pisa_type}")
        pisa_type_lower = str.lower(pisa_type)

        # load Pisa Data
        pisa_file_name = f"20-11-05_PISA_{pisa_type_lower}_total.csv"
        pisa_data = pd.read_csv(path_pisa / pisa_file_name)

        pisa_data = pisa_data.rename(columns={"value": "PISA-SCORE"})
        pisa_data = pisa_data.set_index("Index")

        # load dataset_keys for all data sets that have been prepared up t
o the final state
        datasets_keys = [x.name.replace(".csv", "") for x in path_final_da
ta.iterdir() if x.is_file()]

        # differentiation between different percentages of non-na values
        pbar = tqdm(Iterations)
        for iteration in pbar:
            pbar.set_description(f"{input_data}-{pisa_type}-{iteration}")

           # destinguish between input data types
            if input_data == "aggregated_years":
                data_for_model = create_data_in_pisa_structure_aggregate_y
ears(pisa_data, datasets_keys, iteration, True)
            else:
                data_for_model = create_data_in_pisa_structure_only_pisa_y
ears_no_aggregation(pisa_data, datasets_keys,

                    iteration, True)
            # remove Pisa entries
            data_for_model = remove_PISA_data_from_data(data_for_model)

            # replace nan values and save csv
            data_for_model, column_information = interpolate_nan_values_or
_set_mean_if_not_possible(data_for_model)
            data_for_model.to_csv(Path(path_input/input_data)/f"{pisa_type
}-{iteration}.csv")

            # remove for script - overview of feature Information
            ind = (input_data, pisa_type, iteration)
            overview_input_data.loc[ind, "Inpu_Data_Type"] = input_data
            overview_input_data.loc[ind, "Pisa-Type"] = pisa_type
            overview_input_data.loc[ind, "Percentage_non_NA"] = iteration
            overview_input_data.loc[ind, "Number_Features"] = len(column_i
nformation)-3
            overview_input_data.loc[ind, "Average_Percentage_NAs"] = colum
n_information.iloc[3:, 1].mean()
            overview_input_data.loc[ind, "Average_Percentage_apprximated_I
nterpolation"] = column_information.iloc[3:, 3].mean()
```

```
            overview_input_data.loc[ind, "Average_Percentage_approximated_
Annual_Mean"] = column_information.iloc[3:, 5].mean()


            # store column information - only needed for iterations = 50,
as it includes all features that are also
            # included in the other iterations
            if iteration == 50:
                feature_columns = [x for x in data_for_model.columns if x
!= "PISA-SCORE" and x != "COUNTRY"]
                column_information.to_csv(Path(path_input/input_data) / f"
{pisa_type}-Column_information.csv")

# remove from script
overview_input_data.to_csv(path_input / "Overview_Input_Data_Types.csv")
```

```
aggregated-years:   0%|          | 0/2 [00:00<?, ?it/s]
 0%|          | 0/3 [00:00<?, ?it/s]
aggregated-years-MATH:   0%|          | 0/3 [00:00<?, ?it/s]

 0%|          | 0/11 [00:00<?, ?it/s]

aggregated-years-MATH-50:   0%|          | 0/11 [00:00<?, ?it/s]

aggregated-years-MATH-50:   9%|▉         | 1/11 [00:16<02:49, 16.96s/it]

aggregated-years-MATH-55:   9%|▉         | 1/11 [00:16<02:49, 16.96s/it]

aggregated-years-MATH-55:  18%|█▊        | 2/11 [00:24<02:07, 14.20s/it]

aggregated-years-MATH-60:  18%|█▊        | 2/11 [00:24<02:07, 14.20s/it]

aggregated-years-MATH-60:  27%|██▋       | 3/11 [00:31<01:36, 12.08s/it]

aggregated-years-MATH-65:  27%|██▋       | 3/11 [00:31<01:36, 12.08s/it]

aggregated-years-MATH-65:  36%|███▋      | 4/11 [00:35<01:06,  9.52s/it]

aggregated-years-MATH-70:  36%|███▋      | 4/11 [00:35<01:06,  9.52s/it]

aggregated-years-MATH-70:  45%|████▌     | 5/11 [00:38<00:45,  7.54s/it]

aggregated-years-MATH-75:  45%|████▌     | 5/11 [00:38<00:45,  7.54s/it]

aggregated-years-MATH-75:  55%|█████▍    | 6/11 [00:40<00:30,  6.04s/it]

aggregated-years-MATH-80:  55%|█████▍    | 6/11 [00:40<00:30,  6.04s/it]

aggregated-years-MATH-80:  64%|██████▎   | 7/11 [00:43<00:19,  4.87s/it]

aggregated-years-MATH-85:  64%|██████▎   | 7/11 [00:43<00:19,  4.87s/it]

aggregated-years-MATH-85:  73%|███████▎  | 8/11 [00:43<00:11,  3.69s/it]

aggregated-years-MATH-90:  73%|███████▎  | 8/11 [00:43<00:11,  3.69s/it]

aggregated-years-MATH-90:  82%|████████▏ | 9/11 [00:44<00:05,  2.78s/it]
```

```
aggregated-years-MATH-95:  82%|████████  | 9/11 [00:44<00:05,  2.78s/it]

aggregated-years-MATH-95:  91%|█████████ | 10/11 [00:44<00:02,  2.04s/it]

aggregated-years-MATH-100: 100%|██████████| 11/11 [00:44<00:00,  4.09s/it]

aggregated-years-MATH:  33%|███       | 1/3 [00:45<01:30, 45.01s/it]
aggregated-years-READ:  33%|███       | 1/3 [00:45<01:30, 45.01s/it]

  0%|          | 0/11 [00:00<?, ?it/s]

aggregated-years-READ-50:   0%|          | 0/11 [00:00<?, ?it/s]

aggregated-years-READ-50:   9%|          | 1/11 [00:08<01:29,  8.99s/it]

aggregated-years-READ-55:   9%|          | 1/11 [00:08<01:29,  8.99s/it]

aggregated-years-READ-55:  18%|█         | 2/11 [00:15<01:14,  8.26s/it]

aggregated-years-READ-60:  18%|█         | 2/11 [00:15<01:14,  8.26s/it]

aggregated-years-READ-60:  27%|██        | 3/11 [00:19<00:55,  6.91s/it]

aggregated-years-READ-65:  27%|██        | 3/11 [00:19<00:55,  6.91s/it]

aggregated-years-READ-65:  36%|███       | 4/11 [00:22<00:40,  5.80s/it]

aggregated-years-READ-70:  36%|███       | 4/11 [00:22<00:40,  5.80s/it]

aggregated-years-READ-70:  45%|████      | 5/11 [00:25<00:29,  4.85s/it]

aggregated-years-READ-75:  45%|████      | 5/11 [00:25<00:29,  4.85s/it]

aggregated-years-READ-75:  55%|█████     | 6/11 [00:27<00:19,  3.99s/it]

aggregated-years-READ-80:  55%|█████     | 6/11 [00:27<00:19,  3.99s/it]

aggregated-years-READ-80:  64%|██████    | 7/11 [00:28<00:12,  3.15s/it]

aggregated-years-READ-85:  64%|██████    | 7/11 [00:28<00:12,  3.15s/it]

aggregated-years-READ-85:  73%|███████   | 8/11 [00:29<00:07,  2.45s/it]

aggregated-years-READ-90:  73%|███████   | 8/11 [00:29<00:07,  2.45s/it]

aggregated-years-READ-90:  82%|████████  | 9/11 [00:29<00:03,  1.87s/it]

aggregated-years-READ-95:  82%|████████  | 9/11 [00:29<00:03,  1.87s/it]

aggregated-years-READ-95:  91%|█████████ | 10/11 [00:29<00:01,  1.39s/it]

aggregated-years-READ-100: 100%|██████████| 11/11 [00:29<00:00,  2.72s/it]

aggregated-years-READ:  67%|██████    | 2/3 [01:15<00:40, 40.50s/it]
aggregated-years-SCIENCE:  67%|██████    | 2/3 [01:15<00:40, 40.50s/it]
```

```
  0%|          | 0/11 [00:00<?, ?it/s]

aggregated-years-SCIENCE-50:   0%|          | 0/11 [00:00<?, ?it/s]

aggregated-years-SCIENCE-50:   9%|▉         | 1/11 [00:07<01:15,  7.57s/it]

aggregated-years-SCIENCE-55:   9%|▉         | 1/11 [00:07<01:15,  7.57s/it]

aggregated-years-SCIENCE-55:  18%|█▊        | 2/11 [00:13<01:04,  7.16s/it]

aggregated-years-SCIENCE-60:  18%|█▊        | 2/11 [00:13<01:04,  7.16s/it]

aggregated-years-SCIENCE-60:  27%|██▋       | 3/11 [00:20<00:55,  6.97s/it]

aggregated-years-SCIENCE-65:  27%|██▋       | 3/11 [00:20<00:55,  6.97s/it]

aggregated-years-SCIENCE-65:  36%|███▋      | 4/11 [00:26<00:47,  6.78s/it]

aggregated-years-SCIENCE-70:  36%|███▋      | 4/11 [00:26<00:47,  6.78s/it]

aggregated-years-SCIENCE-70:  45%|████▌     | 5/11 [00:31<00:36,  6.07s/it]

aggregated-years-SCIENCE-75:  45%|████▌     | 5/11 [00:31<00:36,  6.07s/it]

aggregated-years-SCIENCE-75:  55%|█████▍    | 6/11 [00:33<00:25,  5.07s/it]

aggregated-years-SCIENCE-80:  55%|█████▍    | 6/11 [00:33<00:25,  5.07s/it]

aggregated-years-SCIENCE-80:  64%|██████▎   | 7/11 [00:35<00:15,  3.97s/it]

aggregated-years-SCIENCE-85:  64%|██████▎   | 7/11 [00:35<00:15,  3.97s/it]

aggregated-years-SCIENCE-85:  73%|███████▎  | 8/11 [00:36<00:09,  3.04s/it]

aggregated-years-SCIENCE-90:  73%|███████▎  | 8/11 [00:36<00:09,  3.04s/it]

aggregated-years-SCIENCE-90:  82%|████████▏ | 9/11 [00:36<00:04,  2.30s/it]

aggregated-years-SCIENCE-95:  82%|████████▏ | 9/11 [00:36<00:04,  2.30s/it]

aggregated-years-SCIENCE-95:  91%|█████████ | 10/11 [00:36<00:01,  1.70s/it]

aggregated-years-SCIENCE-100: 100%|██████████| 11/11 [00:36<00:00,  3.36s/it]

aggregated-years-SCIENCE: 100%|██████████| 3/3 [01:51<00:00, 37.33s/it]
pisa-years:  50%|█████     | 1/2 [01:52<01:52, 112.00s/it]
  0%|          | 0/3 [00:00<?, ?it/s]
pisa-years-MATH:   0%|          | 0/3 [00:00<?, ?it/s]

  0%|          | 0/11 [00:00<?, ?it/s]

pisa-years-MATH-50:   0%|          | 0/11 [00:00<?, ?it/s]

pisa-years-MATH-50:   9%|▉         | 1/11 [00:21<03:38, 21.89s/it]

pisa-years-MATH-55:   9%|▉         | 1/11 [00:21<03:38, 21.89s/it]
```

```
pisa-years-MATH-55:  18%|          | 2/11 [00:33<02:49, 18.86s/it]
pisa-years-MATH-60:  18%|          | 2/11 [00:33<02:49, 18.86s/it]
pisa-years-MATH-60:  27%|          | 3/11 [00:43<02:08, 16.03s/it]
pisa-years-MATH-65:  27%|          | 3/11 [00:43<02:08, 16.03s/it]
pisa-years-MATH-65:  36%|          | 4/11 [00:49<01:32, 13.25s/it]
pisa-years-MATH-70:  36%|          | 4/11 [00:49<01:32, 13.25s/it]
pisa-years-MATH-70:  45%|          | 5/11 [00:53<01:02, 10.37s/it]
pisa-years-MATH-75:  45%|          | 5/11 [00:53<01:02, 10.37s/it]
pisa-years-MATH-75:  55%|          | 6/11 [00:56<00:40,  8.17s/it]
pisa-years-MATH-80:  55%|          | 6/11 [00:56<00:40,  8.17s/it]
pisa-years-MATH-80:  64%|          | 7/11 [00:58<00:25,  6.41s/it]
pisa-years-MATH-85:  64%|          | 7/11 [00:58<00:25,  6.41s/it]
pisa-years-MATH-85:  73%|          | 8/11 [00:59<00:14,  4.82s/it]
pisa-years-MATH-90:  73%|          | 8/11 [00:59<00:14,  4.82s/it]
pisa-years-MATH-90:  82%|          | 9/11 [01:00<00:07,  3.63s/it]
pisa-years-MATH-95:  82%|          | 9/11 [01:00<00:07,  3.63s/it]
pisa-years-MATH-95:  91%|          | 10/11 [01:01<00:02,  2.67s/it]
pisa-years-MATH-100: 100%|          | 11/11 [01:01<00:00,  5.57s/it]
pisa-years-MATH:  33%|          | 1/3 [01:01<02:02, 61.32s/it]
pisa-years-READ:  33%|          | 1/3 [01:01<02:02, 61.32s/it]
  0%|          | 0/11 [00:00<?, ?it/s]
pisa-years-READ-50:   0%|          | 0/11 [00:00<?, ?it/s]
pisa-years-READ-50:   9%|          | 1/11 [00:24<04:01, 24.19s/it]
pisa-years-READ-55:   9%|          | 1/11 [00:24<04:01, 24.19s/it]
pisa-years-READ-55:  18%|          | 2/11 [00:41<03:19, 22.22s/it]
pisa-years-READ-60:  18%|          | 2/11 [00:41<03:19, 22.22s/it]
pisa-years-READ-60:  27%|          | 3/11 [00:48<02:21, 17.70s/it]
pisa-years-READ-65:  27%|          | 3/11 [00:48<02:21, 17.70s/it]
pisa-years-READ-65:  36%|          | 4/11 [00:53<01:36, 13.78s/it]
```

```
pisa-years-READ-70:  36%|███         | 4/11 [00:53<01:36, 13.78s/it]

pisa-years-READ-70:  45%|████        | 5/11 [00:56<01:03, 10.61s/it]

pisa-years-READ-75:  45%|████        | 5/11 [00:56<01:03, 10.61s/it]

pisa-years-READ-75:  55%|█████       | 6/11 [00:59<00:41,  8.31s/it]

pisa-years-READ-80:  55%|█████       | 6/11 [00:59<00:41,  8.31s/it]

pisa-years-READ-80:  64%|██████      | 7/11 [01:03<00:27,  6.86s/it]

pisa-years-READ-85:  64%|██████      | 7/11 [01:03<00:27,  6.86s/it]

pisa-years-READ-85:  73%|███████     | 8/11 [01:04<00:15,  5.23s/it]

pisa-years-READ-90:  73%|███████     | 8/11 [01:04<00:15,  5.23s/it]

pisa-years-READ-90:  82%|████████    | 9/11 [01:05<00:07,  3.97s/it]

pisa-years-READ-95:  82%|████████    | 9/11 [01:05<00:07,  3.97s/it]

pisa-years-READ-95:  91%|█████████ | 10/11 [01:06<00:02,  2.92s/it]

pisa-years-READ-100: 100%|██████████| 11/11 [01:06<00:00,  6.02s/it]

pisa-years-READ:  67%|██████      | 2/3 [02:07<01:02, 62.81s/it]
pisa-years-SCIENCE:  67%|██████     | 2/3 [02:07<01:02, 62.81s/it]

  0%|          | 0/11 [00:00<?, ?it/s]

pisa-years-SCIENCE-50:   0%|          | 0/11 [00:00<?, ?it/s]

pisa-years-SCIENCE-50:   9%|          | 1/11 [00:21<03:34, 21.42s/it]

pisa-years-SCIENCE-55:   9%|          | 1/11 [00:21<03:34, 21.42s/it]

pisa-years-SCIENCE-55:  18%|█         | 2/11 [00:40<03:05, 20.59s/it]

pisa-years-SCIENCE-60:  18%|█         | 2/11 [00:40<03:05, 20.59s/it]

pisa-years-SCIENCE-60:  27%|██        | 3/11 [00:51<02:21, 17.75s/it]

pisa-years-SCIENCE-65:  27%|██        | 3/11 [00:51<02:21, 17.75s/it]

pisa-years-SCIENCE-65:  36%|███       | 4/11 [00:58<01:43, 14.73s/it]

pisa-years-SCIENCE-70:  36%|███       | 4/11 [00:58<01:43, 14.73s/it]

pisa-years-SCIENCE-70:  45%|████      | 5/11 [01:06<01:14, 12.47s/it]

pisa-years-SCIENCE-75:  45%|████      | 5/11 [01:06<01:14, 12.47s/it]

pisa-years-SCIENCE-75:  55%|█████     | 6/11 [01:11<00:51, 10.32s/it]

pisa-years-SCIENCE-80:  55%|█████     | 6/11 [01:11<00:51, 10.32s/it]
```

```
pisa-years-SCIENCE-80:  64%|██████    | 7/11 [01:14<00:32,  8.21s/it]
pisa-years-SCIENCE-85:  64%|██████    | 7/11 [01:14<00:32,  8.21s/it]
pisa-years-SCIENCE-85:  73%|███████   | 8/11 [01:16<00:18,  6.21s/it]
pisa-years-SCIENCE-90:  73%|███████   | 8/11 [01:16<00:18,  6.21s/it]
pisa-years-SCIENCE-90:  82%|████████  | 9/11 [01:17<00:09,  4.66s/it]
pisa-years-SCIENCE-95:  82%|████████  | 9/11 [01:17<00:09,  4.66s/it]
pisa-years-SCIENCE-95:  91%|█████████ | 10/11 [01:18<00:03,  3.69s/it]
pisa-years-SCIENCE-100: 100%|██████████| 11/11 [01:18<00:00,  7.16s/it]
pisa-years-SCIENCE: 100%|██████████| 3/3 [03:26<00:00, 68.79s/it]
pisa-years: 100%|██████████| 2/2 [05:18<00:00, 159.20s/it]
```

In [15]:
```
# preview of the overall table
overview_input_data
```

Out[15]:

			Inpu_Data_Type	Pisa-Type	Percentage_non_NA	Number_Features	Avera
aggregated-years	**MATH**	**50**	aggregated-years	MATH	50	3136	
		55	aggregated-years	MATH	55	2518	
		60	aggregated-years	MATH	60	1986	
		65	aggregated-years	MATH	65	1480	
		70	aggregated-years	MATH	70	1133	
...	**...**	**...**	...	...	...	...	
pisa-years	**SCIENCE**	**80**	pisa-years	SCIENCE	80	1029	
		85	pisa-years	SCIENCE	85	742	
		90	pisa-years	SCIENCE	90	400	
		95	pisa-years	SCIENCE	95	153	
		100	pisa-years	SCIENCE	100	7	

66 rows × 7 columns

Die verschiedenen Input-Daten sind fertig und gespeichert. Um die bestmöglichen Inputdaten zu wählen haben wir die Modellierung standardisiert.

Unser erstes Modell für Pisa Read inklusive der Tabellen und Plots die wir in Anschluss bei allen

Modellen im Skirpt ausgegeben haben ist hier aufgeführt. Der Random State beim Daten-Split bleibt durchgängig gleich.

In [ ]:
```
from tqdm import tqdm
from pathlib import Path
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.metrics
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, export_graphviz
import pandas as pd
import numpy as np
from Global_configuartions import FOLDERNAME_MODELLING_DATA

# Paths to store results
path_0 = Path(FOLDERNAME_MODELLING_DATA)
path_results = Path(path_0/"Results")
path_tree = Path(path_0/"Trees")
path_input = Path(path_0/"Input Data")

# Defining variables
percentage_test_data = 15
pisa_type = "READ"
iteration = 50 # percentage of non-na data
input_data = "pisa-years"
model_run = "Example"

# create directories
path_results.mkdir(exist_ok=True)
path_results_modelrun = Path(path_results / model_run)
path_results_modelrun.mkdir(exist_ok=True)
path_results_modelrun.mkdir(exist_ok=True)
path_tree.mkdir(exist_ok=True)
path_tree_modelrun = Path(path_tree/model_run)
path_tree_modelrun.mkdir(exist_ok=True)

# load input data and meta data
data_for_model = pd.read_csv(Path(path_input/ input_data)/f"{pisa_type}-{i
teration}.csv")
column_information = pd.read_csv(Path(path_input / input_data) / f"{pisa_t
ype}-Column_information.csv")

if "Unnamed: 0" in column_information.columns:
    column_information = column_information.rename(columns={"Unnamed: 0":
"Index"})
column_information = column_information.set_index("Index")
data_for_model = data_for_model.set_index("Index")
```

In [22]:
```
# prepare Data
feature_columns = [x for x in data_for_model.columns if x != "PISA-SCORE"
and x != "COUNTRY"]
X = data_for_model[feature_columns]
y = data_for_model["PISA-SCORE"]

# Split Data
X_train_final, X_test_final, y_train_final, y_test_final = train_test_spli
```

```
t(X, y, test_size=0.15,

                random_state=42)
X_test_final
```

Out[22]:

Index	YEAR	TABLE_I7 - : Average wage in US dollars based on Purchasing Power Parities [Units US Dollar]	TABLE_I7 - : Average wage in national currency units [Units National currency]	TABLE_I7 - : Personal income tax & employee social security contributions (All-in rate) [Units Percentage]	TABLE_I7 - : Personal income tax [Units Percentage]	TABLE_I7 - : Threshold (expressed as a multiple of the average wage) [Units Ratio]	TABLE_I7 - : Top tax rates [Units Percentage]	AIR_E Co nan [
CAN-2009	2009	34971.943384	4.246170e+04	46.409600	46.409600	2.928406	46.409600	1
MEX-2006	2006	9991.225020	7.114901e+04	24.740414	22.663747	1.494245	30.000000	1
RUS-2006	2006	30437.884470	1.389916e+06	44.792793	38.573877	2.425805	42.126980	
MEX-2018	2018	13391.304684	1.196008e+05	35.000000	35.000000	27.161727	35.000000	1
POL-2012	2012	21729.823250	3.909317e+04	38.106883	21.380417	2.574819	32.000000	1
USA-2015	2015	50310.150000	5.031015e+04	48.600000	46.250000	8.228397	46.250000	3
LVA-2006	2006	9122.244575	4.194000e+03	26.669106	17.669106	0.117571	25.000000	
ISR-2018	2018	40208.667481	1.503460e+05	50.000000	50.000000	4.455957	50.000000	
AUT-2006	2006	39515.572694	3.440517e+04	43.743843	43.743843	2.005014	43.714286	
LVA-2018	2018	23130.219925	1.129600e+04	26.787500	16.037500	2.381327	27.200000	
BRA-2012	2012	38150.130798	1.730470e+06	43.837414	38.561723	3.836157	41.514821	
POL-2018	2018	29837.181667	5.207780e+04	39.715268	21.889408	1.938253	32.000000	1
EST-2006	2006	13254.331342	6.782805e+03	12.789949	11.989949	0.204639	23.833333	
NZL-2015	2015	37813.428294	5.517167e+04	33.000000	33.000000	1.269253	33.000000	
NLD-2015	2015	60532.620619	4.888167e+04	63.615443	50.996354	1.226701	52.000000	

PER-2015	2015	41659.416174	1.953567e+06	45.217031	40.502895	5.442282	42.867377	
SWE-2006	2006	34363.892808	3.184283e+05	56.585000	56.585000	1.468876	56.585000	
AUS-2018	2018	57942.780024	8.434081e+04	48.000000	48.000000	2.134886	48.000000	
PRT-2018	2018	31478.267087	1.813758e+04	59.641467	48.641467	15.492044	54.653333	
GRC-2009	2009	34232.509168	2.422335e+04	49.096000	33.096000	3.686712	40.000000	
HUN-2015	2015	23748.921685	3.093252e+06	34.500000	16.000000	0.261211	16.000000	
IDN-2009	2009	34582.971707	1.600212e+06	43.664086	37.570475	2.785071	40.870308	
BEL-2018	2018	62027.482277	4.791950e+04	60.268752	46.068662	1.047282	52.995083	
NZL-2006	2006	26641.585609	4.004300e+04	39.000000	39.000000	1.499096	39.000000	
MEX-2000	2000	7964.329832	4.860652e+04	42.930000	40.000000	48.589815	40.000000	2
CHL-2018	2018	22950.317567	9.446798e+06	35.833333	35.833333	8.054252	35.833333	1
ESP-2012	2012	36225.969268	2.558321e+04	48.166667	48.166667	8.638392	48.166667	
CHL-2015	2015	21519.658925	8.102029e+06	40.000000	40.000000	10.212185	40.000000	2
JPN-2012	2012	45544.254938	4.849311e+06	47.770833	47.229167	4.668594	50.000000	1
BEL-2006	2006	41920.231718	3.701190e+04	59.278511	45.112323	1.047091	53.500000	
SVN-2009	2009	24623.380656	1.573328e+04	54.039000	31.939000	1.423781	41.000000	
SGP-2015	2015	41659.416174	1.953567e+06	45.217031	40.502895	5.442282	42.867377	
ITA-2003	2003	27544.456337	2.280017e+04	50.643675	42.152008	3.394068	46.066667	1
CAN-2018	2018	43774.094412	5.260736e+04	53.529600	53.529600	4.184406	53.529600	1
NOR-2012	2012	54533.378095	4.945742e+05	47.800000	40.000000	1.571126	40.000000	
GRC-2000	2000	23089.457852	1.545890e+04	52.798875	36.898875	3.664760	43.875000	
POL-2015	2015	25408.493083	4.484767e+04	38.751185	20.925325	2.247580	32.000000	1

AUT-2012	2012	48555.293775	4.000236e+04	43.714286	43.714286	2.023822	43.714286	
HUN-2012	2012	21714.823886	2.720630e+06	32.690000	14.773333	0.261211	18.666667	

39 rows × 29391 columns

In [20]:
```
%%time

# create and train model
model_tree = DecisionTreeRegressor()
model_tree.fit(X_train_final, y_train_final)

# predict data
y_predict = model_tree.predict(X_test_final)
```

```
Wall time: 10.6 s
```

In [28]:
```
# add country information and Pisa Score to X_test for plot
test_plot = X_test_final.copy()
test_plot = test_plot.merge(data_for_model.loc[X_test_final.index, "COUNTRY"], how="left", left_index=True, right_index=True)
test_plot = test_plot.merge(y_test_final, how="left", left_index=True, right_index=True)
test_plot = test_plot.sort_values(["YEAR"])
test_plot
```

Out[28]:

Index	YEAR	TABLE_I7 - : Average wage in US dollars based on Purchasing Power Parities [Units US Dollar]	TABLE_I7 - : Average wage in national currency units [Units National currency]	TABLE_I7 - : Personal income tax & employee social security contributions (All-in rate) [Units Percentage]	TABLE_I7 - : Personal income tax [Units Percentage]	TABLE_I7 - : Threshold (expressed as a multiple of the average wage) [Units Ratio]	TABLE_I7 - : Top tax rates [Units Percentage]	AIR_E M Cor nan [T
GRC-2000	2000	23089.457852	1.545890e+04	52.798875	36.898875	3.664760	43.875000	
MEX-2000	2000	7964.329832	4.860652e+04	42.930000	40.000000	48.589815	40.000000	2(
ITA-2003	2003	27544.456337	2.280017e+04	50.643675	42.152008	3.394068	46.066667	1(
MEX-2006	2006	9991.225020	7.114901e+04	24.740414	22.663747	1.494245	30.000000	1!
RUS-2006	2006	30437.884470	1.389916e+06	44.792793	38.573877	2.425805	42.126980	
LVA-	2006	9122.244575	4.194000e+03	26.669106	17.669106	0.117571	25.000000	

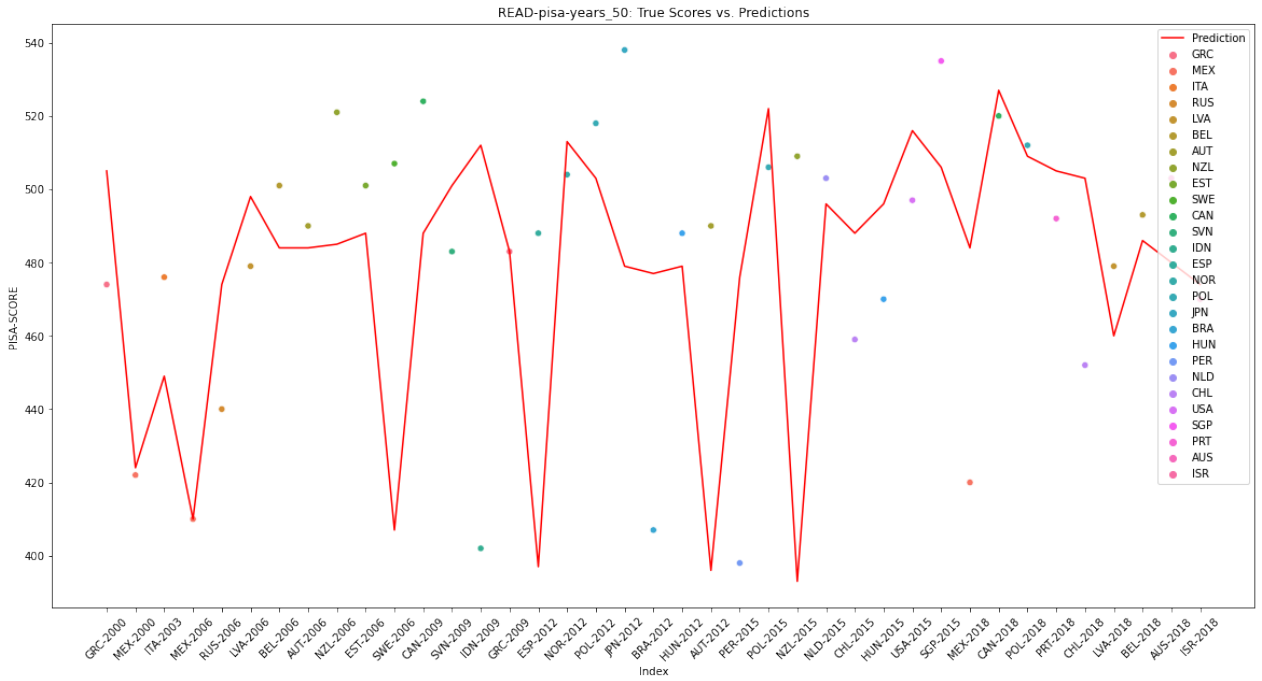
2006								
BEL-2006	2006	41920.231718	3.701190e+04	59.278511	45.112323	1.047091	53.500000	2
AUT-2006	2006	39515.572694	3.440517e+04	43.743843	43.743843	2.005014	43.714286	4
NZL-2006	2006	26641.585609	4.004300e+04	39.000000	39.000000	1.499096	39.000000	1
EST-2006	2006	13254.331342	6.782805e+03	12.789949	11.989949	0.204639	23.833333	
SWE-2006	2006	34363.892808	3.184283e+05	56.585000	56.585000	1.468876	56.585000	1
CAN-2009	2009	34971.943384	4.246170e+04	46.409600	46.409600	2.928406	46.409600	19
SVN-2009	2009	24623.380656	1.573328e+04	54.039000	31.939000	1.423781	41.000000	
IDN-2009	2009	34582.971707	1.600212e+06	43.664086	37.570475	2.785071	40.870308	8
GRC-2009	2009	34232.509168	2.422335e+04	49.096000	33.096000	3.686712	40.000000	1
ESP-2012	2012	36225.969268	2.558321e+04	48.166667	48.166667	8.638392	48.166667	6
NOR-2012	2012	54533.378095	4.945742e+05	47.800000	40.000000	1.571126	40.000000	1
POL-2012	2012	21729.823250	3.909317e+04	38.106883	21.380417	2.574819	32.000000	19
JPN-2012	2012	45544.254938	4.849311e+06	47.770833	47.229167	4.668594	50.000000	10
BRA-2012	2012	38150.130798	1.730470e+06	43.837414	38.561723	3.836157	41.514821	1
HUN-2012	2012	21714.823886	2.720630e+06	32.690000	14.773333	0.261211	18.666667	3
AUT-2012	2012	48555.293775	4.000236e+04	43.714286	43.714286	2.023822	43.714286	4
PER-2015	2015	41659.416174	1.953567e+06	45.217031	40.502895	5.442282	42.867377	6
POL-2015	2015	25408.493083	4.484767e+04	38.751185	20.925325	2.247580	32.000000	10
NZL-2015	2015	37813.428294	5.517167e+04	33.000000	33.000000	1.269253	33.000000	1
NLD-2015	2015	60532.620619	4.888167e+04	63.615443	50.996354	1.226701	52.000000	1
CHL-2015	2015	21519.658925	8.102029e+06	40.000000	40.000000	10.212185	40.000000	20
HUN-								

2015	2015	23748.921685	3.093252e+06	34.500000	16.000000	0.261211	16.000000	
USA-2015	2015	50310.150000	5.031015e+04	48.600000	46.250000	8.228397	46.250000	3
SGP-2015	2015	41659.416174	1.953567e+06	45.217031	40.502895	5.442282	42.867377	
MEX-2018	2018	13391.304684	1.196008e+05	35.000000	35.000000	27.161727	35.000000	1
CAN-2018	2018	43774.094412	5.260736e+04	53.529600	53.529600	4.184406	53.529600	1
POL-2018	2018	29837.181667	5.207780e+04	39.715268	21.889408	1.938253	32.000000	1
PRT-2018	2018	31478.267087	1.813758e+04	59.641467	48.641467	15.492044	54.653333	
CHL-2018	2018	22950.317567	9.446798e+06	35.833333	35.833333	8.054252	35.833333	1
LVA-2018	2018	23130.219925	1.129600e+04	26.787500	16.037500	2.381327	27.200000	
BEL-2018	2018	62027.482277	4.791950e+04	60.268752	46.068662	1.047282	52.995083	
AUS-2018	2018	57942.780024	8.434081e+04	48.000000	48.000000	2.134886	48.000000	
ISR-2018	2018	40208.667481	1.503460e+05	50.000000	50.000000	4.455957	50.000000	

39 rows × 29393 columns

In [29]:
```
###Prediction, True Scores ###
plt.figure(iteration,figsize=(20,10))
sc_plot_predict = sns.scatterplot(test_plot.index, test_plot["PISA-SCORE"]
,
                                  hue=test_plot["COUNTRY"])
plt.xticks(rotation=45)
plt.plot(y_predict, c="r", label = "Prediction")
plt.legend(loc=1)
plt.title(f"{pisa_type}-{input_data}_{iteration}: True Scores vs. Predicti
ons")
plt.show()
plt.close()
```

```
C:\Users\Nina\anaconda3\envs\Seminar-MaschinellesLernen\lib\site-packages\
seaborn\_decorators.py:36: FutureWarning: Pass the following variables as
keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```



In [31]:
```
# store model-run data
results = X_test_final.copy()
results = results.drop(columns=feature_columns)
results["Modelrun"] = model_run
results = results.merge(data_for_model[["COUNTRY", "YEAR"]], on="Index")
results["PISA-SCORE"] = y_test_final
results["y_predict"] = y_predict
results["Absoloute Error"] = abs(results["PISA-SCORE"]-results["y_predict"
])
results["Squared Error"] = abs(results["PISA-SCORE"]-results["y_predict"])
**2
results["RMSE"] = np.sqrt(results["Squared Error"])
results["R^2"] = np.nan
results
```

Out[31]:

Index	Modelrun	COUNTRY	YEAR	PISA-SCORE	y_predict	Absoloute Error	Squared Error	RMSE	R^2
CAN-2009	Example	CAN	2009	524.0	505.0	19.0	361.0	19.0	NaN
MEX-2006	Example	MEX	2006	410.0	424.0	14.0	196.0	14.0	NaN
RUS-2006	Example	RUS	2006	440.0	449.0	9.0	81.0	9.0	NaN
MEX-2018	Example	MEX	2018	420.0	410.0	10.0	100.0	10.0	NaN
POL-2012	Example	POL	2012	518.0	474.0	44.0	1936.0	44.0	NaN
USA-2015	Example	USA	2015	497.0	498.0	1.0	1.0	1.0	NaN

LVA-2006	Example	LVA	2006	479.0	484.0	5.0	25.0	5.0	NaN
ISR-2018	Example	ISR	2018	470.0	484.0	14.0	196.0	14.0	NaN
AUT-2006	Example	AUT	2006	490.0	485.0	5.0	25.0	5.0	NaN
LVA-2018	Example	LVA	2018	479.0	488.0	9.0	81.0	9.0	NaN
BRA-2012	Example	BRA	2012	407.0	407.0	0.0	0.0	0.0	NaN
POL-2018	Example	POL	2018	512.0	488.0	24.0	576.0	24.0	NaN
EST-2006	Example	EST	2006	501.0	501.0	0.0	0.0	0.0	NaN
NZL-2015	Example	NZL	2015	509.0	512.0	3.0	9.0	3.0	NaN
NLD-2015	Example	NLD	2015	503.0	483.0	20.0	400.0	20.0	NaN
PER-2015	Example	PER	2015	398.0	397.0	1.0	1.0	1.0	NaN
SWE-2006	Example	SWE	2006	507.0	513.0	6.0	36.0	6.0	NaN
AUS-2018	Example	AUS	2018	503.0	503.0	0.0	0.0	0.0	NaN
PRT-2018	Example	PRT	2018	492.0	479.0	13.0	169.0	13.0	NaN
GRC-2009	Example	GRC	2009	483.0	477.0	6.0	36.0	6.0	NaN
HUN-2015	Example	HUN	2015	470.0	479.0	9.0	81.0	9.0	NaN
IDN-2009	Example	IDN	2009	402.0	396.0	6.0	36.0	6.0	NaN
BEL-2018	Example	BEL	2018	493.0	476.0	17.0	289.0	17.0	NaN
NZL-2006	Example	NZL	2006	521.0	522.0	1.0	1.0	1.0	NaN
MEX-2000	Example	MEX	2000	422.0	393.0	29.0	841.0	29.0	NaN
CHL-2018	Example	CHL	2018	452.0	496.0	44.0	1936.0	44.0	NaN
ESP-2012	Example	ESP	2012	488.0	488.0	0.0	0.0	0.0	NaN
CHL-2015	Example	CHL	2015	459.0	496.0	37.0	1369.0	37.0	NaN

JPN-2012	Example	JPN	2012	538.0	516.0	22.0	484.0	22.0	NaN
BEL-2006	Example	BEL	2006	501.0	506.0	5.0	25.0	5.0	NaN
SVN-2009	Example	SVN	2009	483.0	484.0	1.0	1.0	1.0	NaN
SGP-2015	Example	SGP	2015	535.0	527.0	8.0	64.0	8.0	NaN
ITA-2003	Example	ITA	2003	476.0	509.0	33.0	1089.0	33.0	NaN
CAN-2018	Example	CAN	2018	520.0	505.0	15.0	225.0	15.0	NaN
NOR-2012	Example	NOR	2012	504.0	503.0	1.0	1.0	1.0	NaN
GRC-2000	Example	GRC	2000	474.0	460.0	14.0	196.0	14.0	NaN
POL-2015	Example	POL	2015	506.0	486.0	20.0	400.0	20.0	NaN
AUT-2012	Example	AUT	2012	490.0	480.0	10.0	100.0	10.0	NaN
HUN-2012	Example	HUN	2012	488.0	474.0	14.0	196.0	14.0	NaN

In [32]:
```
# overall information about model
new_row_df = pd.DataFrame(columns=["Modelrun", "COUNTRY", "PISA-SCORE", "y_predict", "Absoloute Error", "Squared Error", "RMSE", "R^2"],
                          data=[[model_run, np.nan, np.nan, np.nan, sklearn.metrics.mean_absolute_error(y_test_final, y_predict),
                           sklearn.metrics.mean_squared_error(y_test_final, y_predict),
                           np.sqrt(sklearn.metrics.mean_squared_error(y_test_final, y_predict)),
                           model_tree.score(X_test_final, y_test_final)]], index=[model_run])
new_row_df
```

Out[32]:

	Modelrun	COUNTRY	PISA-SCORE	y_predict	Absoloute Error	Squared Error	RMSE	R^2
Example	Example	NaN	NaN	NaN	12.538462	296.487179	17.218803	0.781063

In [33]:
```
results_with_total_overview = pd.concat([new_row_df, results])
results_with_total_overview.loc[model_run, "Percentage Test-Data"] = percentage_test_data
results_with_total_overview.loc[model_run, "Optimal_Depth"] = model_tree.get_depth()
results_with_total_overview.loc[model_run, "R2 Training Data"] = model_tre
```

```
e.score(X_train_final, y_train_final)
results_with_total_overview
```

Out[33]:

	ModelRun	COUNTRY	PISA-SCORE	y_predict	Absoloute Error	Squared Error	RMSE	R^2	YE
Example	Example	NaN	NaN	NaN	12.538462	296.487179	17.218803	0.781063	N
CAN-2009	Example	CAN	524.0	505.0	19.000000	361.000000	19.000000	NaN	200
MEX-2006	Example	MEX	410.0	424.0	14.000000	196.000000	14.000000	NaN	200
RUS-2006	Example	RUS	440.0	449.0	9.000000	81.000000	9.000000	NaN	200
MEX-2018	Example	MEX	420.0	410.0	10.000000	100.000000	10.000000	NaN	201
POL-2012	Example	POL	518.0	474.0	44.000000	1936.000000	44.000000	NaN	201
USA-2015	Example	USA	497.0	498.0	1.000000	1.000000	1.000000	NaN	201
LVA-2006	Example	LVA	479.0	484.0	5.000000	25.000000	5.000000	NaN	200
ISR-2018	Example	ISR	470.0	484.0	14.000000	196.000000	14.000000	NaN	201
AUT-2006	Example	AUT	490.0	485.0	5.000000	25.000000	5.000000	NaN	200
LVA-2018	Example	LVA	479.0	488.0	9.000000	81.000000	9.000000	NaN	201
BRA-2012	Example	BRA	407.0	407.0	0.000000	0.000000	0.000000	NaN	201
POL-2018	Example	POL	512.0	488.0	24.000000	576.000000	24.000000	NaN	201
EST-2006	Example	EST	501.0	501.0	0.000000	0.000000	0.000000	NaN	200
NZL-2015	Example	NZL	509.0	512.0	3.000000	9.000000	3.000000	NaN	201
NLD-2015	Example	NLD	503.0	483.0	20.000000	400.000000	20.000000	NaN	201
PER-2015	Example	PER	398.0	397.0	1.000000	1.000000	1.000000	NaN	201
SWE-2006	Example	SWE	507.0	513.0	6.000000	36.000000	6.000000	NaN	200
AUS-2018	Example	AUS	503.0	503.0	0.000000	0.000000	0.000000	NaN	201
PRT-	Example	PRT	492.0	479.0	13.000000	169.000000	13.000000	NaN	201

2018									
GRC-2009	Example	GRC	483.0	477.0	6.000000	36.000000	6.000000	NaN	2009
HUN-2015	Example	HUN	470.0	479.0	9.000000	81.000000	9.000000	NaN	2015
IDN-2009	Example	IDN	402.0	396.0	6.000000	36.000000	6.000000	NaN	2009
BEL-2018	Example	BEL	493.0	476.0	17.000000	289.000000	17.000000	NaN	2018
NZL-2006	Example	NZL	521.0	522.0	1.000000	1.000000	1.000000	NaN	2006
MEX-2000	Example	MEX	422.0	393.0	29.000000	841.000000	29.000000	NaN	2000
CHL-2018	Example	CHL	452.0	496.0	44.000000	1936.000000	44.000000	NaN	2018
ESP-2012	Example	ESP	488.0	488.0	0.000000	0.000000	0.000000	NaN	2012
CHL-2015	Example	CHL	459.0	496.0	37.000000	1369.000000	37.000000	NaN	2015
JPN-2012	Example	JPN	538.0	516.0	22.000000	484.000000	22.000000	NaN	2012
BEL-2006	Example	BEL	501.0	506.0	5.000000	25.000000	5.000000	NaN	2006
SVN-2009	Example	SVN	483.0	484.0	1.000000	1.000000	1.000000	NaN	2009
SGP-2015	Example	SGP	535.0	527.0	8.000000	64.000000	8.000000	NaN	2015
ITA-2003	Example	ITA	476.0	509.0	33.000000	1089.000000	33.000000	NaN	2003
CAN-2018	Example	CAN	520.0	505.0	15.000000	225.000000	15.000000	NaN	2018
NOR-2012	Example	NOR	504.0	503.0	1.000000	1.000000	1.000000	NaN	2012
GRC-2000	Example	GRC	474.0	460.0	14.000000	196.000000	14.000000	NaN	2000
POL-2015	Example	POL	506.0	486.0	20.000000	400.000000	20.000000	NaN	2015
AUT-2012	Example	AUT	490.0	480.0	10.000000	100.000000	10.000000	NaN	2012
HUN-2012	Example	HUN	488.0	474.0	14.000000	196.000000	14.000000	NaN	2012

In [39]:
```
# get tree information - iterate through the nodes of the Tree
```

```
# merge with information about the feature
# store as representation of tree
total_nodes = model_tree.tree_.node_count
features = pd.DataFrame()
feature_columns = list(X.columns)
for node_id in range(0, total_nodes):
    if model_tree.tree_.feature[node_id] == -2: # id of all Leaf nodes
        features.loc[node_id, "Feature"] = "Leaf"
        features.loc[node_id, "Threshold"] = np.nan
    else:
        features.loc[node_id, "Feature"] = feature_columns[model_tree.tree
_.feature[node_id]]
        features.loc[node_id, "Threshold"] = model_tree.tree_.threshold[no
de_id]
    features.loc[node_id, "Samples"] = model_tree.tree_.n_node_samples[nod
e_id]
    features.loc[node_id, "Value"] = model_tree.tree_.value[node_id][0]
features_merged = features.merge(column_information, how="left", left_on="
Feature", right_index=True)
features_merged = features_merged.set_index("Feature")
features_merged
```

Out[39]:

Feature	Threshold	Samples	Value	NA-Values	NA-Values [%]	Inpolated	Inpolated [%]	Ar
GREEN_GROWTH - : Energy intensity, TPES per capita [Units Tonnes of oil equivalent (toe)]	1.840521	221.0	486.619910	4.0	1.538462	1.0	0.384615	
LFS_SEXAGE_I_R - : Men + 20 to 24 + Employment/population ratio [Units Percentage]	61.571884	29.0	418.931034	6.0	2.307692	1.0	0.384615	
NRR - : Couple without children - partner's earnings: 67% of the AW + 9 + Average Wage + No + No []	65.500000	8.0	455.250000	81.0	31.153846	32.0	12.307692	
USLHRS_D - : Men + 20 to 24 + Total employment + 1 to 19 hours [Thousands Persons]	17.500000	4.0	444.750000	49.0	18.846154	28.0	10.769231	
STLABOUR - Level, rate or quantity series, s.a. : Unemployed population, Aged 15 and over, Males [Thousands Persons]	485.119400	2.0	441.500000	96.0	36.923077	0.0	0.000000	
...	...	...	...	...	...	...	...	

Leaf	NaN	6.0	508.833333	NaN	NaN	NaN	NaN
Leaf	NaN	13.0	501.307692	NaN	NaN	NaN	NaN
TENURE_DIS - : Women + 15 to 19 + Total employment + Total [Thousands Employed]	67.919704	22.0	496.636364	95.0	36.538462	47.0	18.076923
Leaf	NaN	1.0	483.000000	NaN	NaN	NaN	NaN
Leaf	NaN	21.0	497.285714	NaN	NaN	NaN	NaN

155 rows × 9 columns

In [45]:
```
# create Robustness Information
# Identify how often the different features appear in relation to the trai
ning samples in the tree
# sort according to samples - importance of that feature pursumed by the t
otal number of samples at the feature
robust_check = features.drop(columns=["Threshold", "Value"], axis=1)
robust_check = robust_check.groupby("Feature").sum() # sum up the number o
f samples at nodes of the feature
robust_check = robust_check.sort_values("Samples", ascending=False)
robust_check = robust_check.drop("Leaf") # drop leaf node - not relevant t
o feature analysis
robust_check = robust_check.merge(column_information, how="left", left_on=
"Feature", right_index=True)

# calculate how affected the samples passing through the node are by the a
pproximation of  na values by the annual mean
robust_check["Robustness - Samples affected by NA [Samples]"] = robust_che
ck["Samples"] * robust_check["NA-Values [%]"] / 100
robust_check["Robustness - Samples affected by Annual Mean [Samples]"] = r
obust_check["Samples"] * robust_check["Annual Mean [%]"] / 100
robust_check
```

Out[45]:

Feature	Samples	NA-Values	NA-Values [%]	Inpolated	Inpolated [%]	Annual Mean	Annual Mean [%]	Robustnes - Sample affected b N [Sample
GREEN_GROWTH - : Energy intensity, TPES per capita [Units Tonnes of oil equivalent (toe)]	221.0	4.0	1.538462	1.0	0.384615	3.0	1.153846	3.40000
EAG_TRANS - Share of population by education and labour force status Value : Men + 15-24	192.0	56.0	21.538462	37.0	14.230769	19.0	7.307692	41.35384

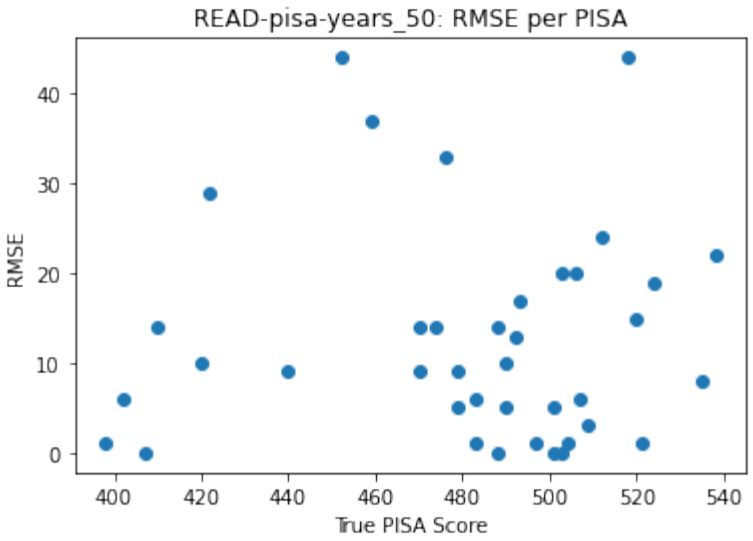
years + Other employed [Units]								
TIM_2019_MAIN - Domestic compensation of employees content of gross exports : France + Fabricated metal products [Millions US Dollar]	116.0	113.0	43.461538	96.0	36.923077	17.0	6.538462	50.41538
TIM_2019_MAIN - Share of domestic employment embodied in exports of final products : World + Mining support service activities [Units Percentage]	76.0	110.0	42.307692	103.0	39.615385	7.0	2.692308	32.1538
TIM_2019_MAIN - Domestic compensation of employees content of gross exports : Finland + Total business sector services [Millions US Dollar]	67.0	109.0	41.923077	99.0	38.076923	10.0	3.846154	28.0884
...	...	...	...	...	...	...	...	
PTRUB - : Couple with 2 children - partner's earnings: Minimum Wage + 2 + Average Wage + No + No + Yes []	2.0	120.0	46.153846	29.0	11.153846	91.0	35.000000	0.9230
NRR - : Couple with 2 children - partner's earnings: 67% of the AW + 8 + Minimum Wage + Yes + No []	2.0	121.0	46.538462	31.0	11.923077	90.0	34.615385	0.9307
IOTSI4_2018 - : Domestic output and imports + Telecommunications + Chemicals and pharmaceutical products [Millions US Dollar]	2.0	105.0	40.384615	102.0	39.230769	3.0	1.153846	0.8076
HEALTH_PHMC - % of total sales : N05B-Anxiolytics []	2.0	115.0	44.230769	33.0	12.692308	82.0	31.538462	0.8846
NRR - : Couple without children - partner's earnings:								

67% of the AW + 11 + Average Wage + Yes + No []	2.0	80.0	30.769231	32.0	12.307692	48.0	18.461538	0.61538

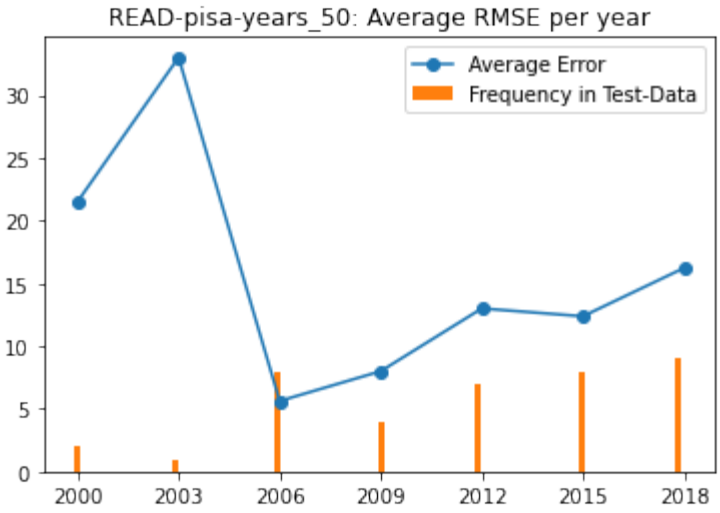
77 rows × 9 columns

In [ ]:
```
###Export trees both with and without feature names####
export_graphviz(model_tree, out_file=f"{pisa_type}-{iteration}_Tree_{model
_run}_with_names.dot", feature_names=feature_columns)
export_graphviz(model_tree, out_file=f"{pisa_type}-{iteration}_Tree_{model
_run}.dot")
```

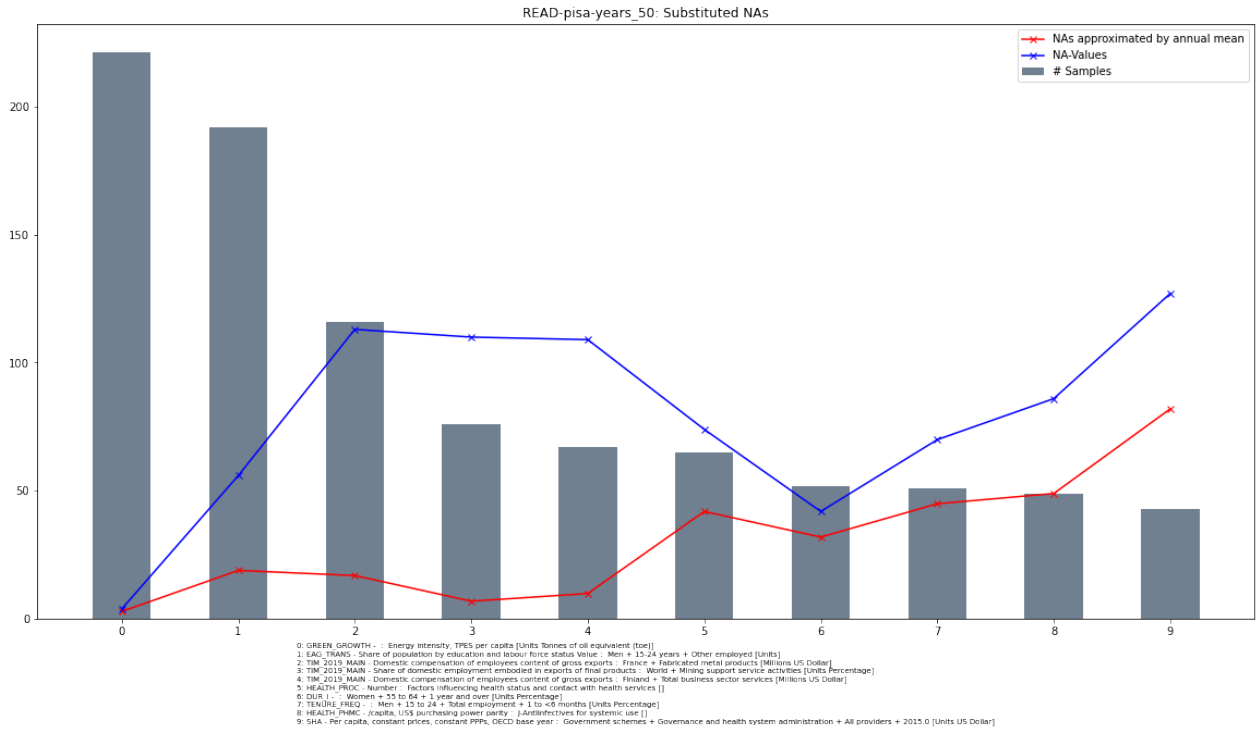
In [46]:
```
### Show abs. error per pisa score ###
# see whether there is potential for improvement in specific areas of the
pisa score
plt.figure()
plt.scatter(results["PISA-SCORE"], results["RMSE"])
plt.xlabel("True PISA Score")
plt.ylabel("RMSE")
plt.title(f"{pisa_type}-{input_data}_{iteration}: RMSE per PISA")
plt.show()
plt.close()
```



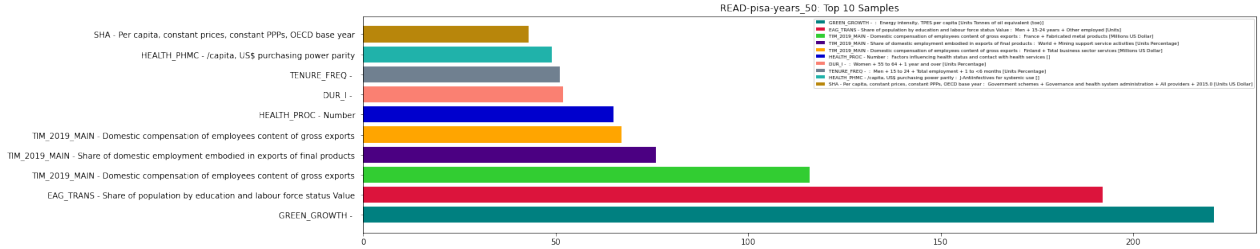
In [48]:
```
### Average Error per year ###
# identify systematic over/underprediction on yearly basis taking the freq
uency of the year's appearance into account
plt.figure()
tmp = results[["YEAR", "RMSE"]].groupby("YEAR").mean()
plt.plot(tmp.index, tmp, marker="o", label="Average Error")
plt.hist(results["YEAR"], bins=100, align="left", label="Frequency in Test
-Data")
plt.xticks(tmp.index)
plt.legend()
plt.title(f"{pisa_type}-{input_data}_{iteration}: Average RMSE per year")
plt.show()
plt.close()
```



In [68]:
```
# display robustness of the best features - show how many na values they h
ad and how many of the values were substituted by the annual mean
plt.figure(figsize=(20, 20))
data = robust_check[:10].copy()
feature_names = robust_check.index.to_list()[:10]
feature_names = [f"{x}: {feature_names[x]}" for x in range(len(feature_nam
es))]
feature_names = "\n".join(feature_names)

plt.plot(range(len(data)), data["Annual Mean"], marker="x", c="r", label="
NAs approximated by annual mean")
plt.plot(range(len(data)), data["NA-Values"], marker="x", c="b", label="NA
-Values")
plt.hist(data.index, bins=range(len(data)+1), color="slategray", align="le
ft", rwidth=0.5,
         weights=data["Samples"], label="# Samples")
plt.legend()
plt.xticks(range(len(data.index)), range(len(data)));
plt.xlabel(feature_names, fontsize="x-small", ma="left")
plt.gcf().subplots_adjust(bottom=0.5)
plt.title(f"{pisa_type}-{input_data}_{iteration}: Substituted NAs")
plt.show()
plt.close()
```

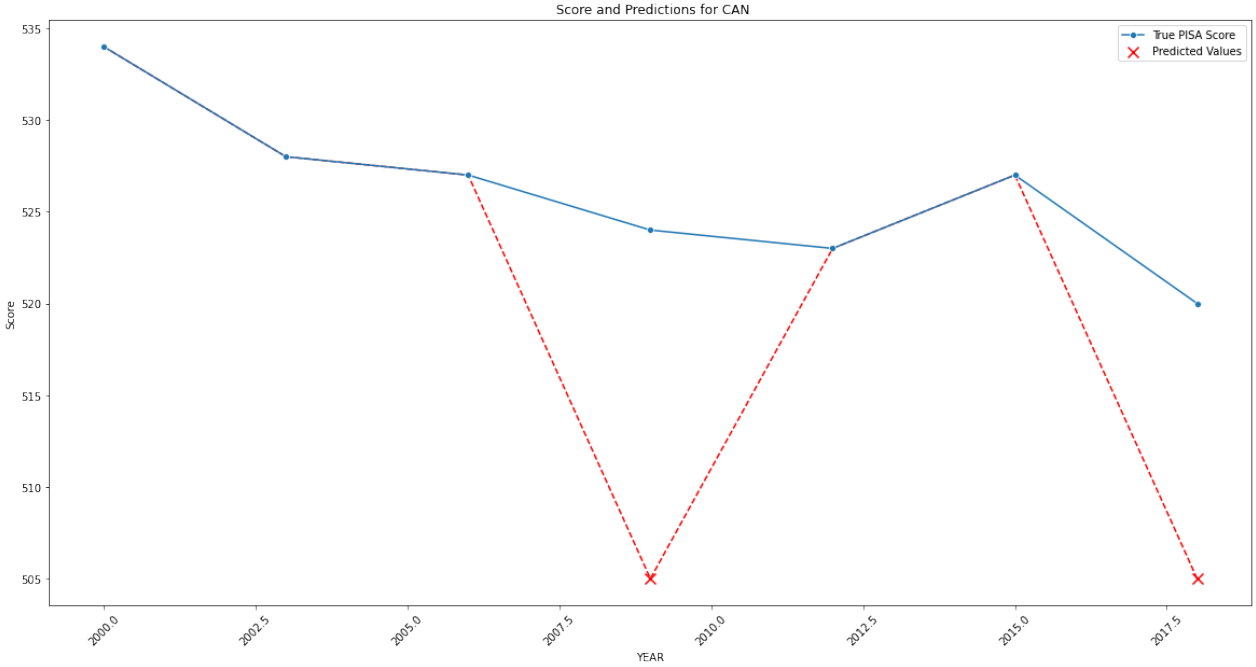


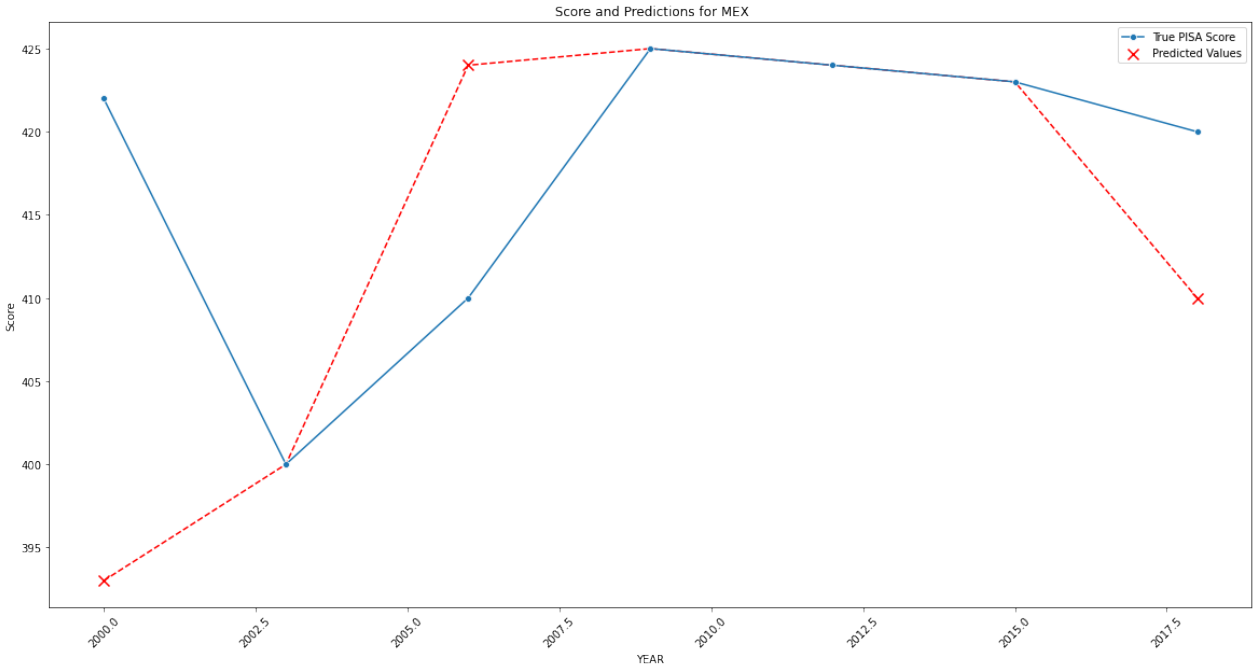
In [70]:
```
# get overview of features with most samples in tree
colors = ["teal", "crimson", "limegreen", "indigo", "orange", "mediumblue"
, "salmon",
                        "slategray", "lightseagreen", "darkgoldenrod"]
data["name"] = [x.split(":")[0] for x in data.index]
data["labels"] = [str(x) for x in range(len(data) )]
data["color"] = colors[:len(data)]

fig = plt.figure(figsize=(20, 5))
for f in data.index:
    plt.barh(data.loc[f, "labels"], data.loc[f, "Samples"], align="center"
,
             tick_label=data.loc[f, "name"], color=data.loc[f, "color"])

plt.title(f"{pisa_type}-{input_data}_{iteration}: Top 10 Samples")
plt.legend(labels=data.index, ncol=1, bbox_transform=fig.transFigure, loc=
"upper right",
           fontsize='xx-small')
plt.yticks(ticks=np.arange(len(data)), labels=data["name"])
plt.show()
plt.close()
```

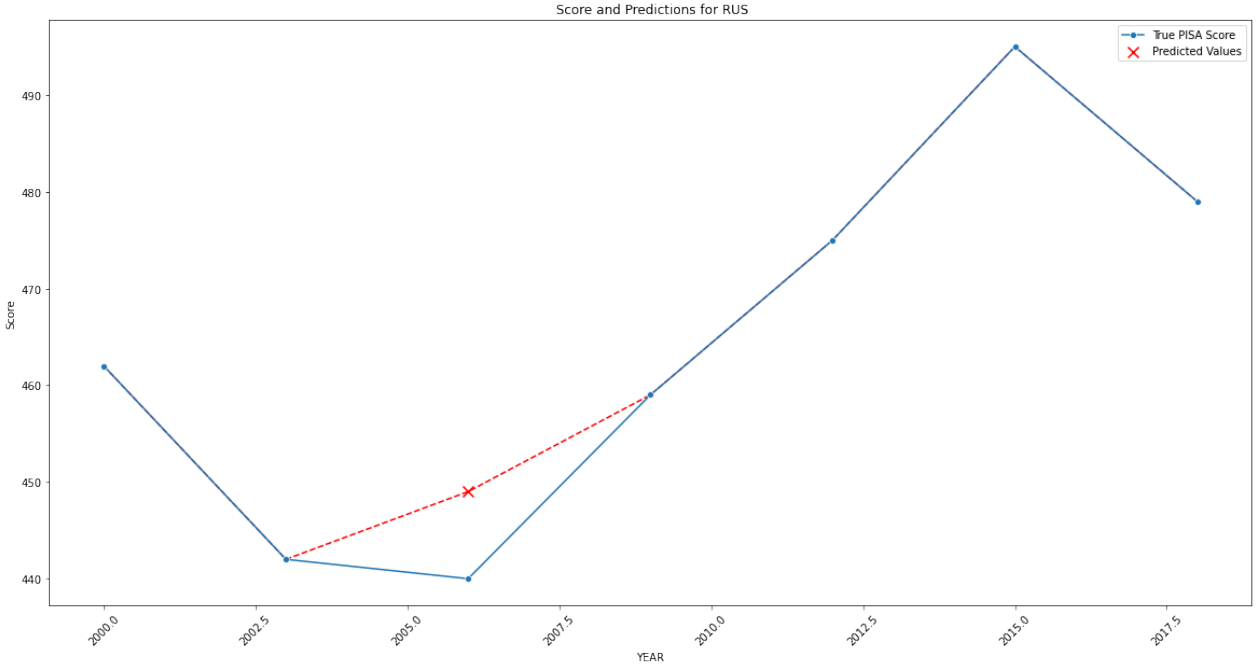


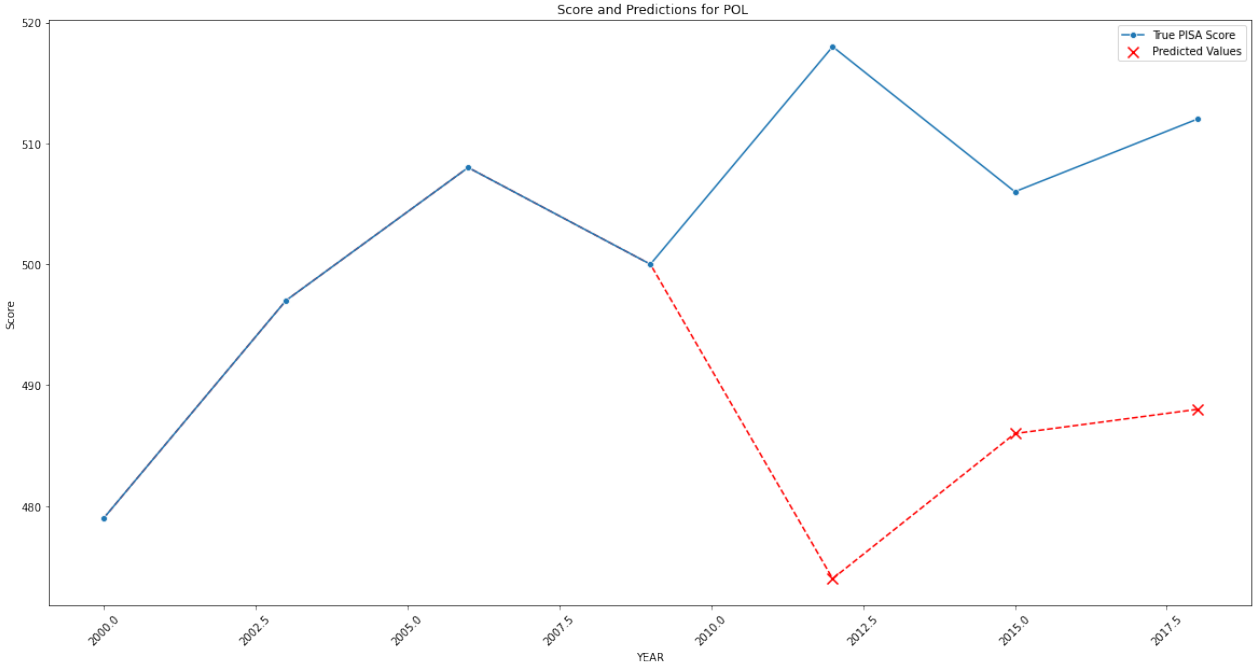
In [71]:
```
# plot predictions for countries to sidentify systematic over-underpredict
ion on country level
estimated_countries = list(results["COUNTRY"].unique())
```

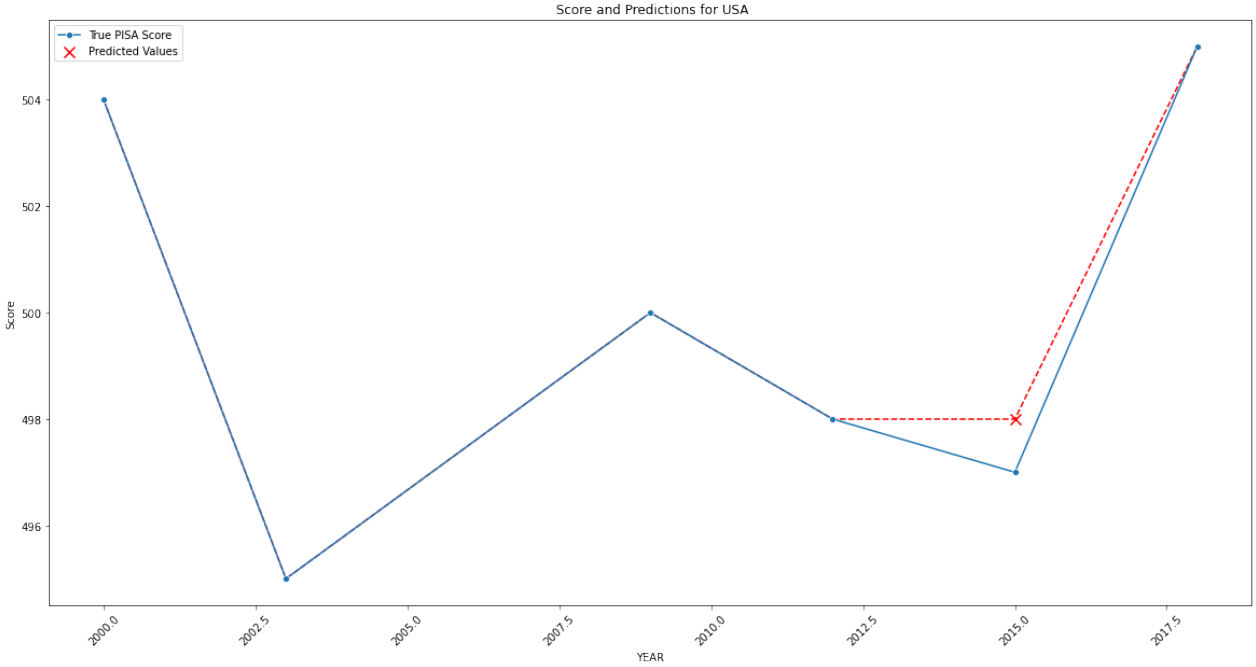
```
estimated_countries_data = data_for_model.loc[data_for_model["COUNTRY"].is
in(estimated_countries),
                                              ["PISA-SCORE", "YEAR", "COUNT
RY"]]

estimated_countries_data = estimated_countries_data.merge(results["y_predi
ct"], how="left", left_index=True,
                                                          right_index= True)
estimated_countries_data["new_pisa"] = estimated_countries_data["y_predict
"]
estimated_countries_data["new_pisa"] = estimated_countries_data["y_predict
"].fillna(estimated_countries_data["PISA-SCORE"])
estimated_countries_data = estimated_countries_data.sort_values(["YEAR"])

years = list(estimated_countries_data["YEAR"].unique())

for country in estimated_countries:

    tmp = estimated_countries_data.loc[estimated_countries_data["COUNTRY"]
== country].copy()

    plt.figure(figsize=(20,10))


    s = sns.lineplot(data = tmp, x="YEAR", y="new_pisa", color='r', linest
yle='--')
    p = sns.lineplot(data = tmp, x="YEAR", y="PISA-SCORE", marker='o', lab
el="True PISA Score")
    plt.scatter(tmp["YEAR"], tmp["y_predict"], color='r', marker='x', labe
l="Predicted Values", s=100)
    plt.xticks(rotation=45)
    plt.ylabel("Score")
    plt.legend()
    plt.title(f" Score and Predictions for {country}")
    p.get_xaxis().get_major_formatter().set_useOffset(False)
    plt.show()
    plt.close()
```

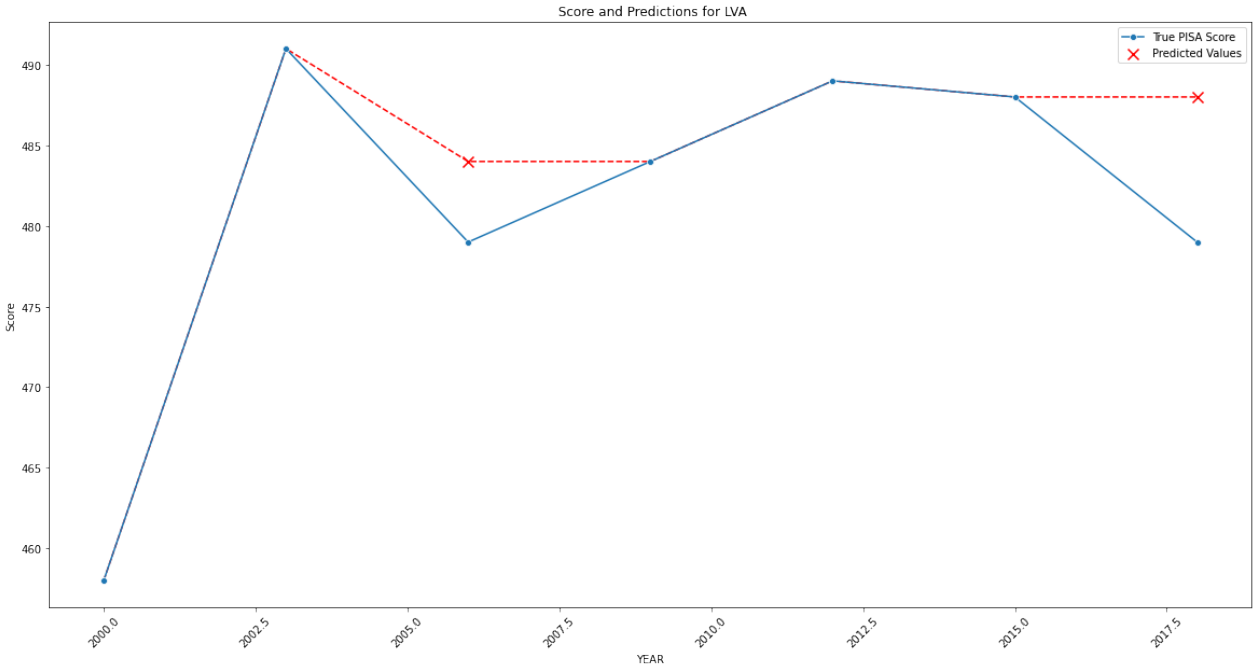

Score and Predictions for CAN
True PISA Score
Predicted Values
Score
YEAR
535
530
525
520
515
510
505
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

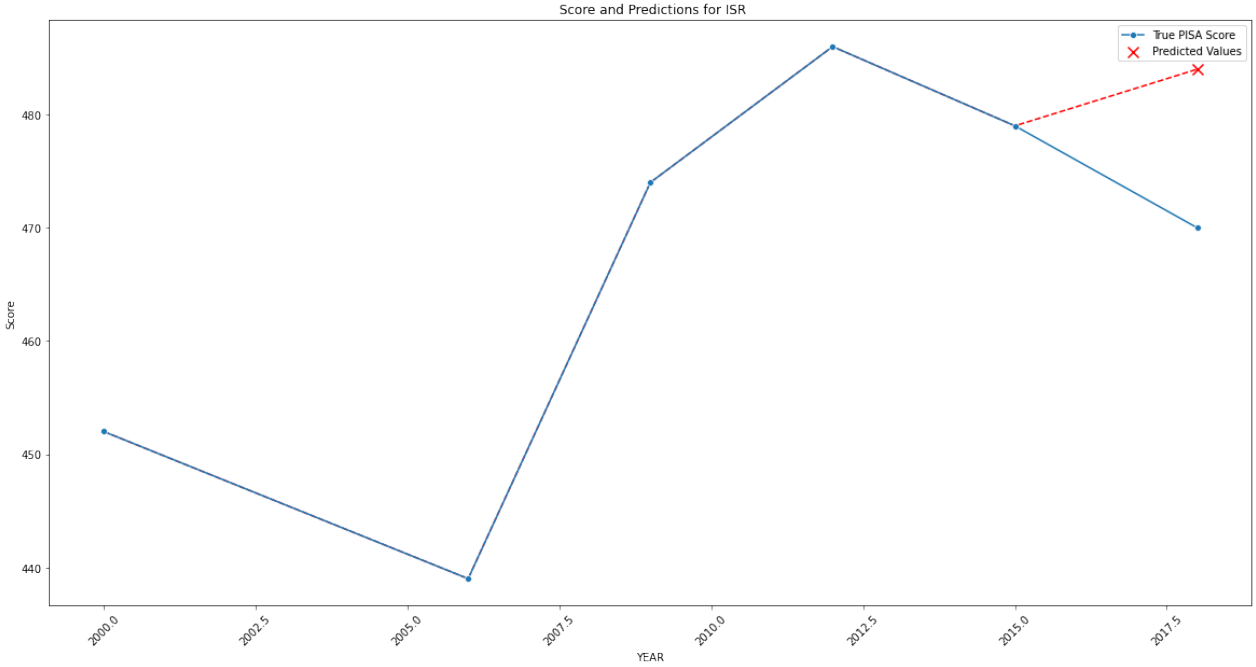

Score and Predictions for MEX
True PISA Score
Predicted Values
Score
YEAR
425
420
415
410
405
400
395
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

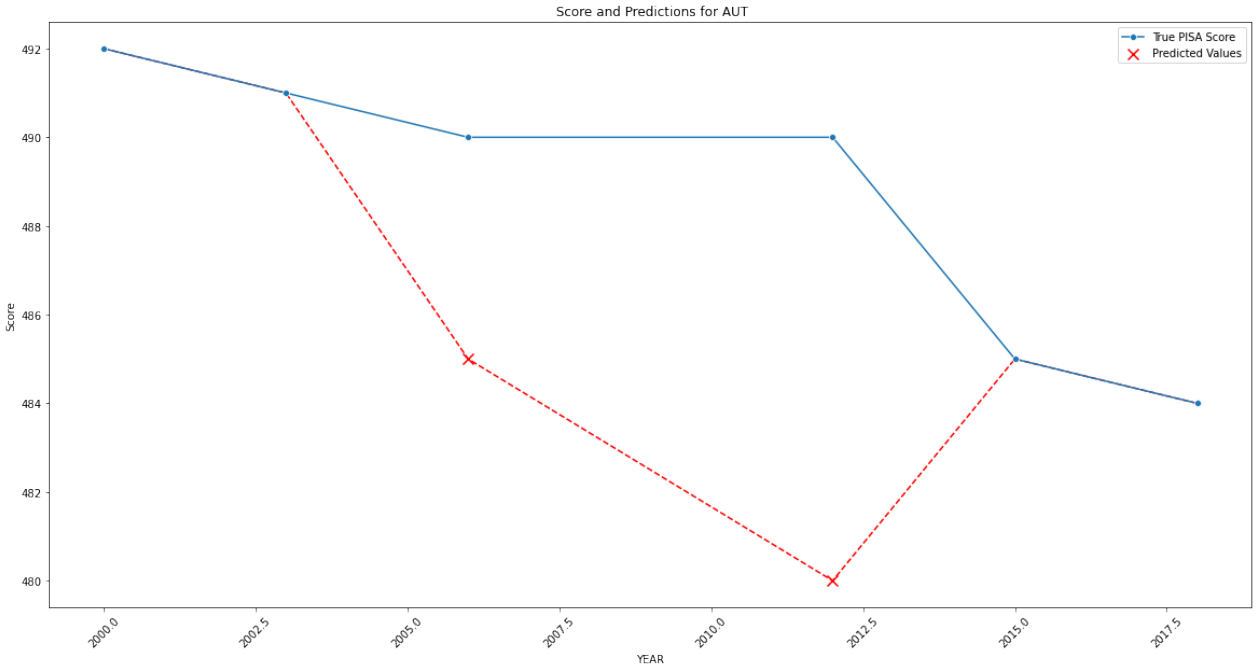

Score and Predictions for RUS
True PISA Score
Predicted Values
Score
YEAR
490
480
470
460
450
440
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

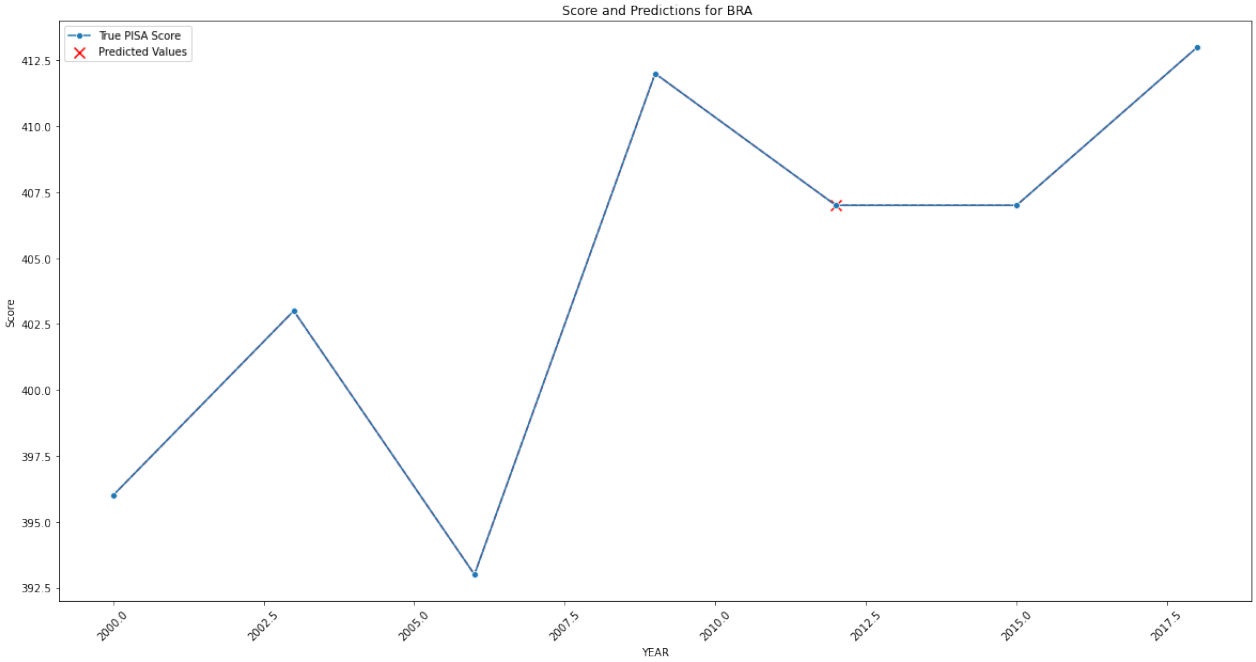

Score and Predictions for POL
True PISA Score
Predicted Values
Score
YEAR
520
510
500
490
480
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

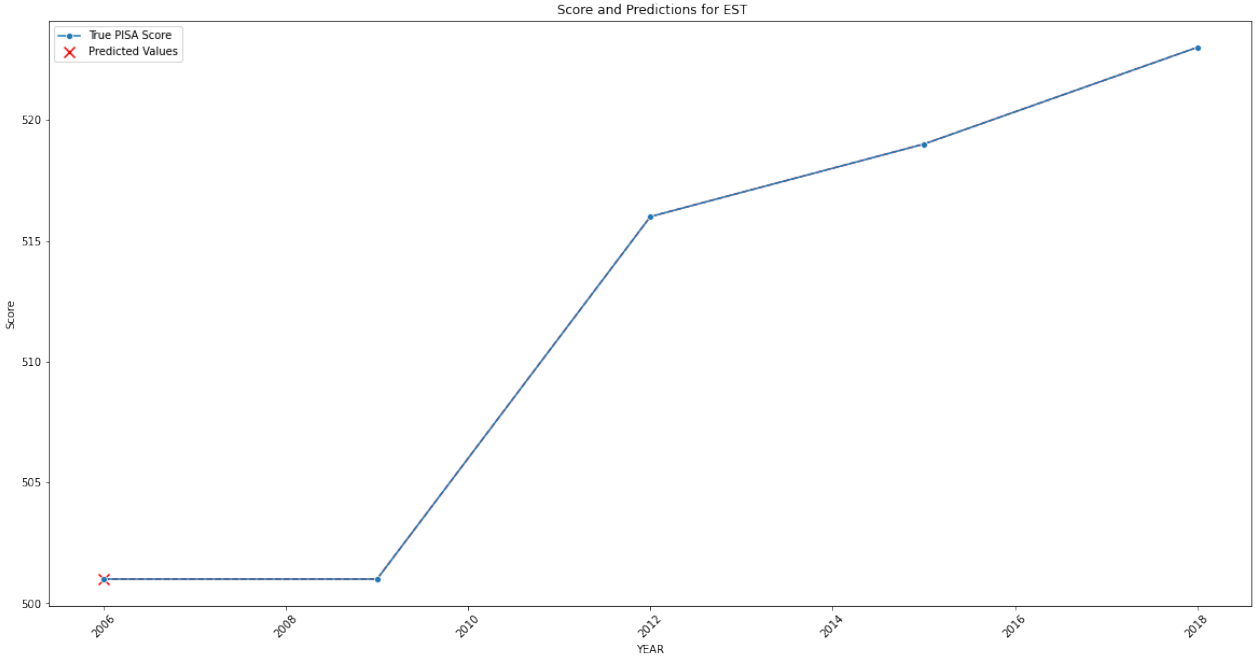

Score and Predictions for USA
True PISA Score
Predicted Values
Score
YEAR
504
502
500
498
496
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

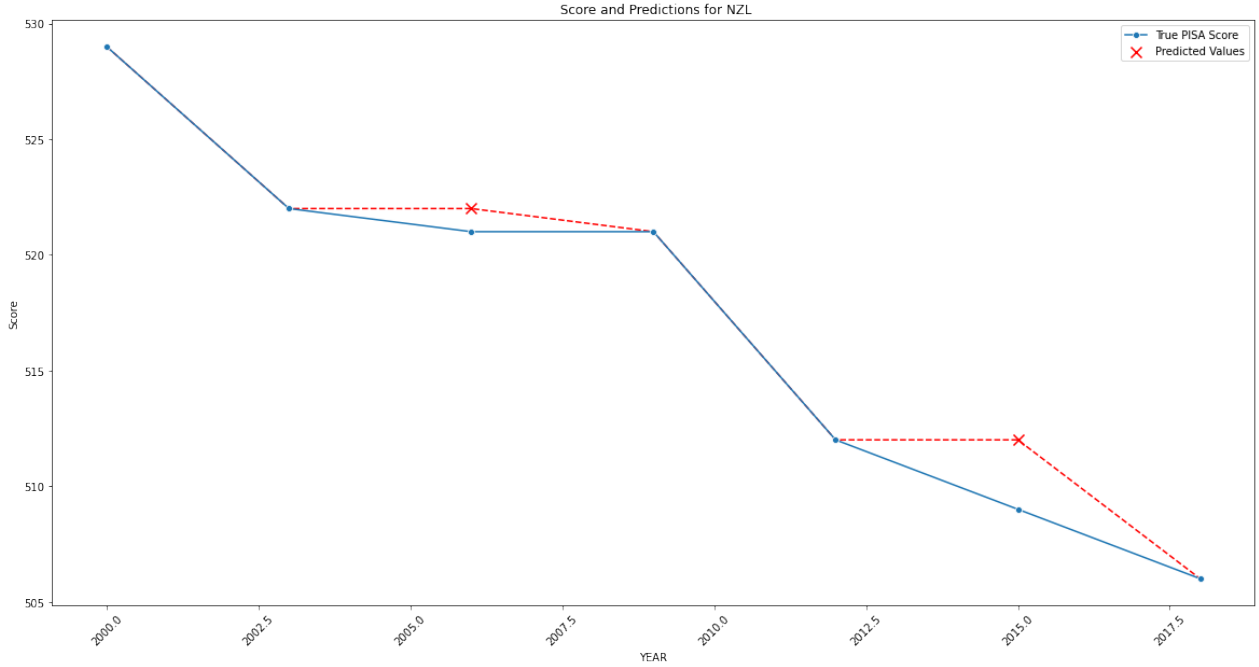

Score and Predictions for LVA
True PISA Score
Predicted Values
Score
YEAR
490
485
480
475
470
465
460
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

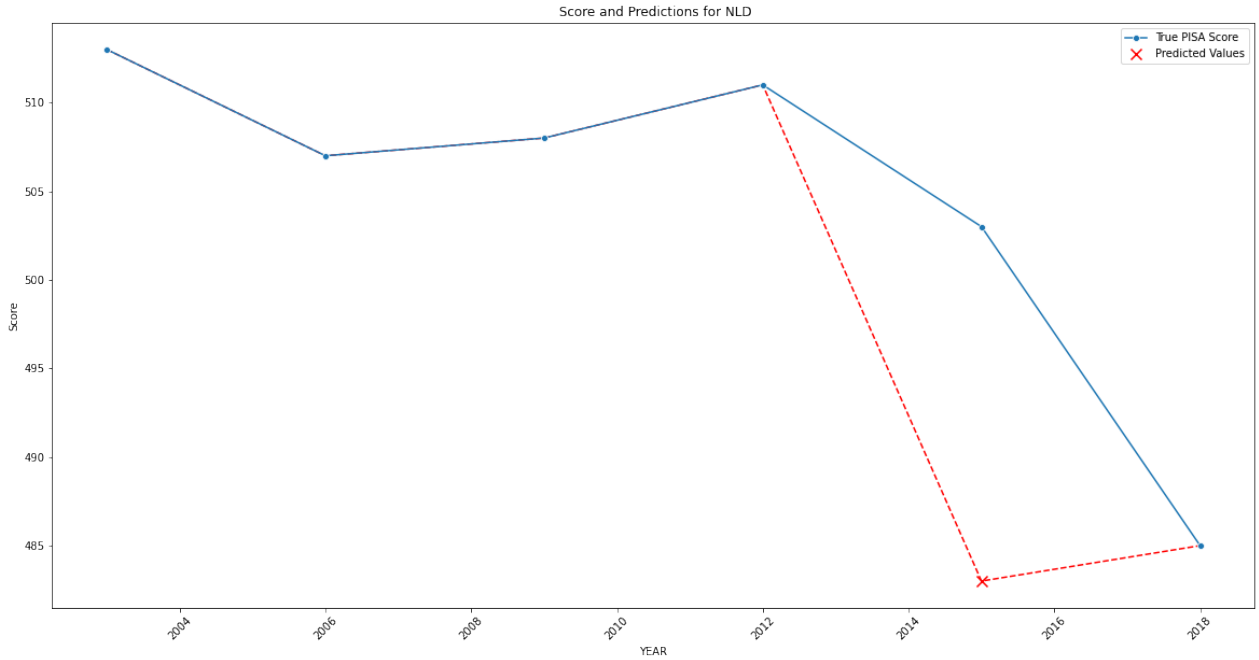

Score and Predictions for ISR
True PISA Score
Predicted Values
Score
480
470
460
450
440
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
YEAR

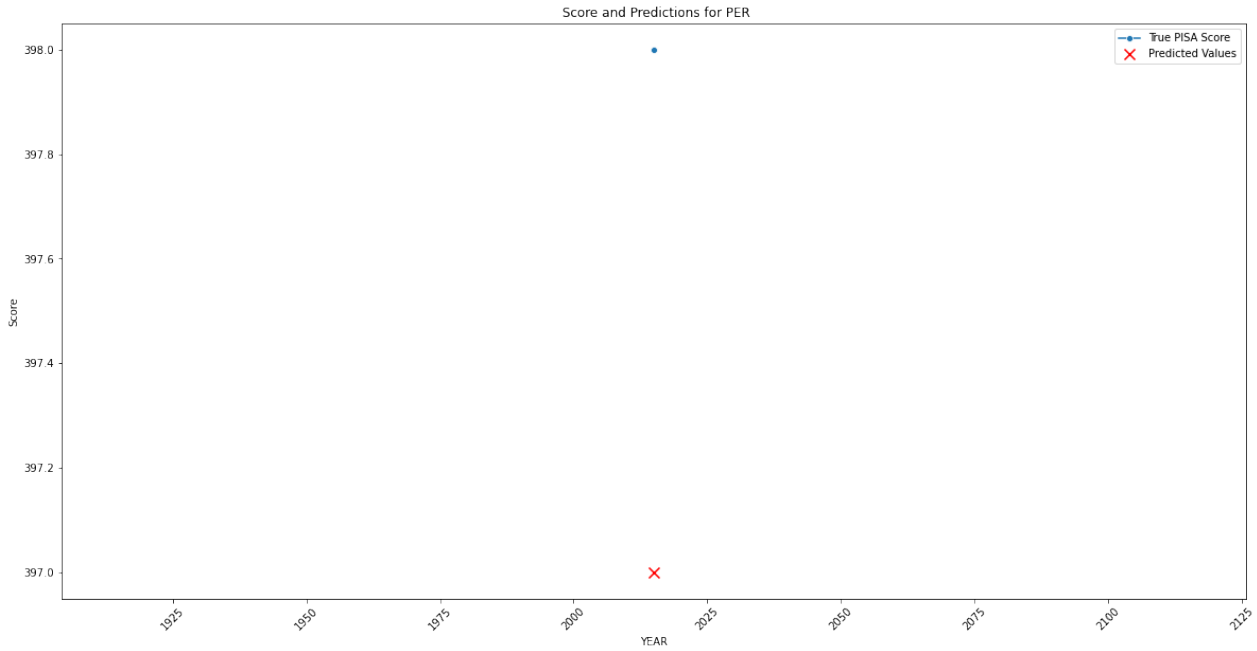

Score and Predictions for AUT
True PISA Score
Predicted Values
Score
492
490
488
486
484
482
480
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
YEAR

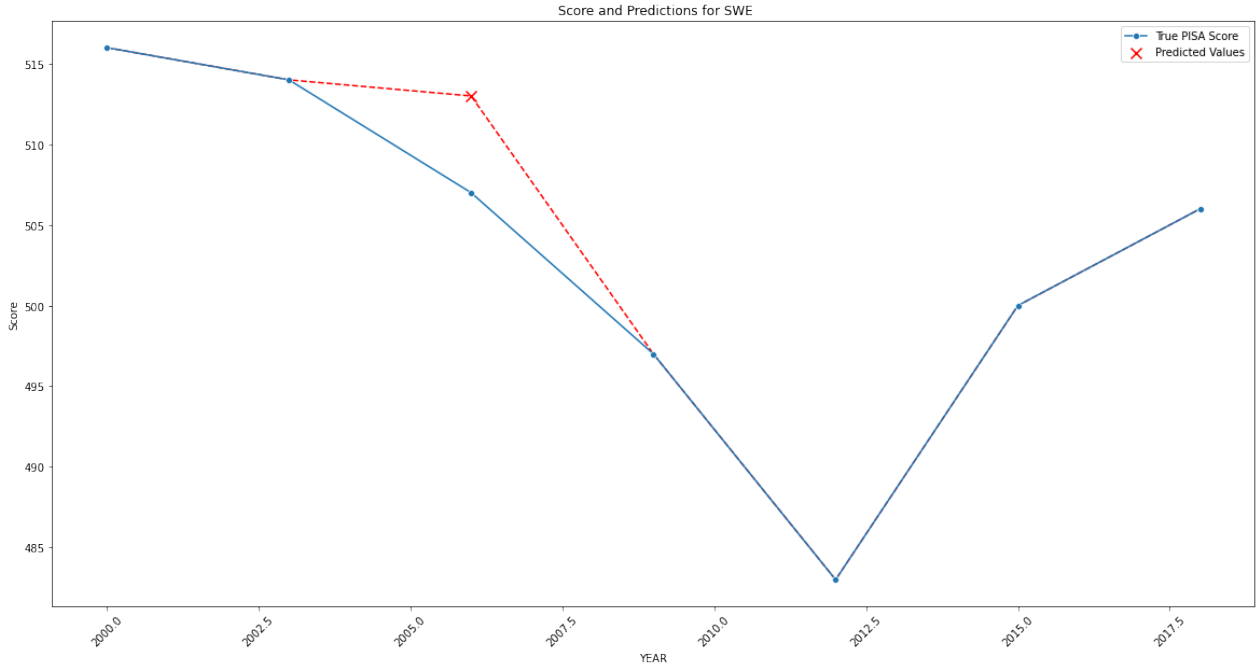

Score and Predictions for BRA
True PISA Score
Predicted Values
Score
YEAR
412.5
410.0
407.5
405.0
402.5
400.0
397.5
395.0
392.5
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

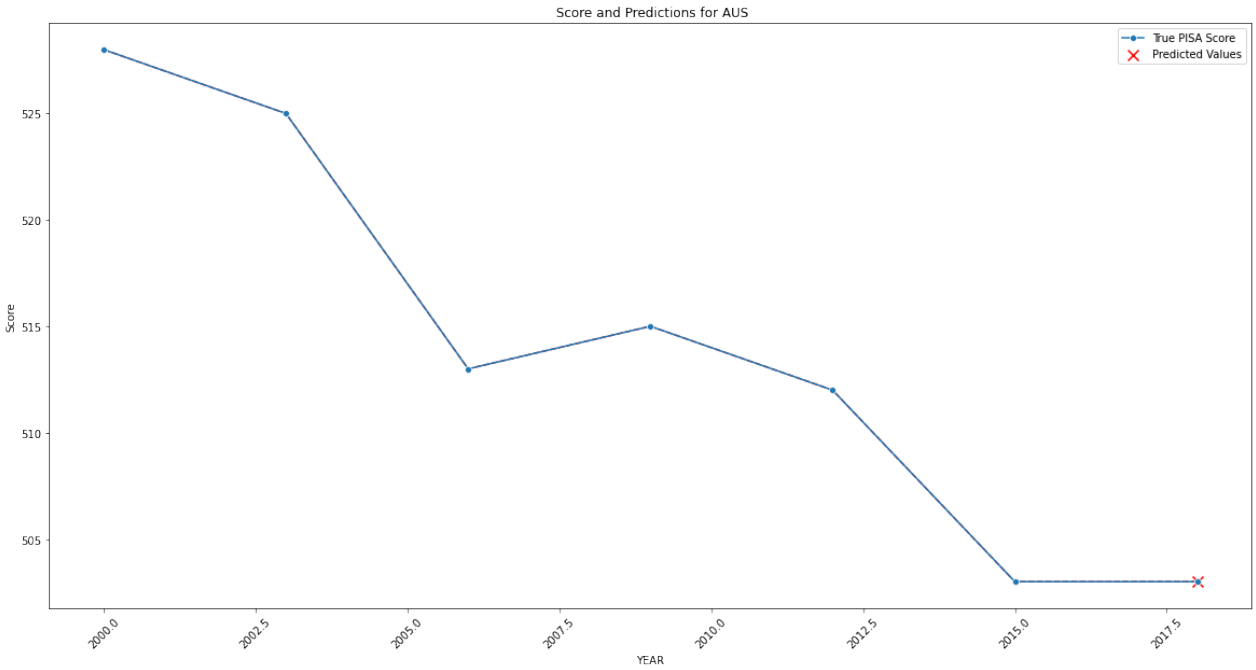

Score and Predictions for EST
True PISA Score
Predicted Values
Score
YEAR
520
515
510
505
500
2006
2008
2010
2012
2014
2016
2018

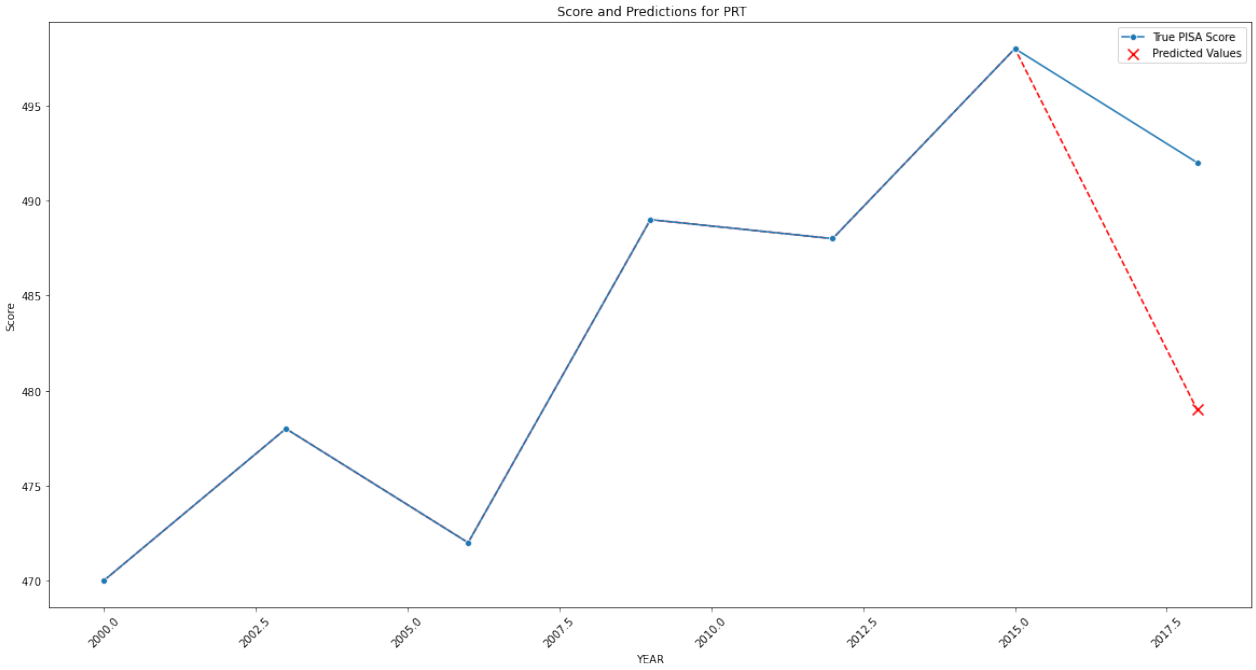

Score and Predictions for NZL
True PISA Score
Predicted Values
Score
YEAR
530
525
520
515
510
505
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

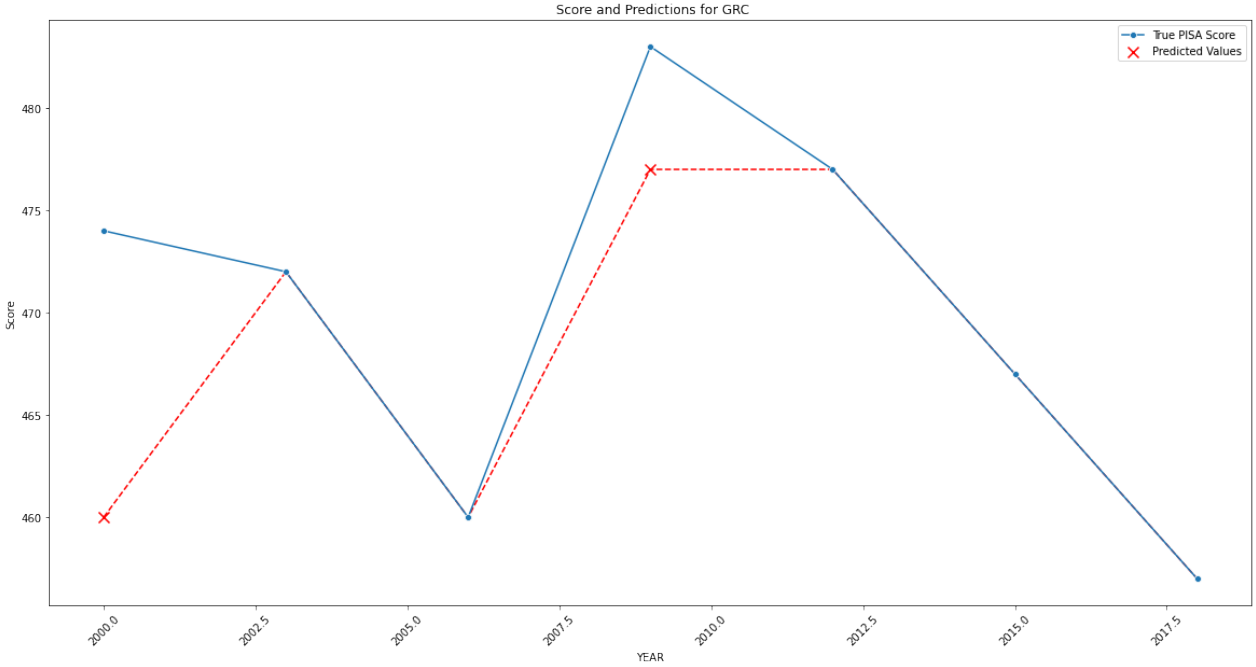

Score and Predictions for NLD
True PISA Score
Predicted Values
Score
YEAR
510
505
500
495
490
485
2004
2006
2008
2010
2012
2014
2016
2018

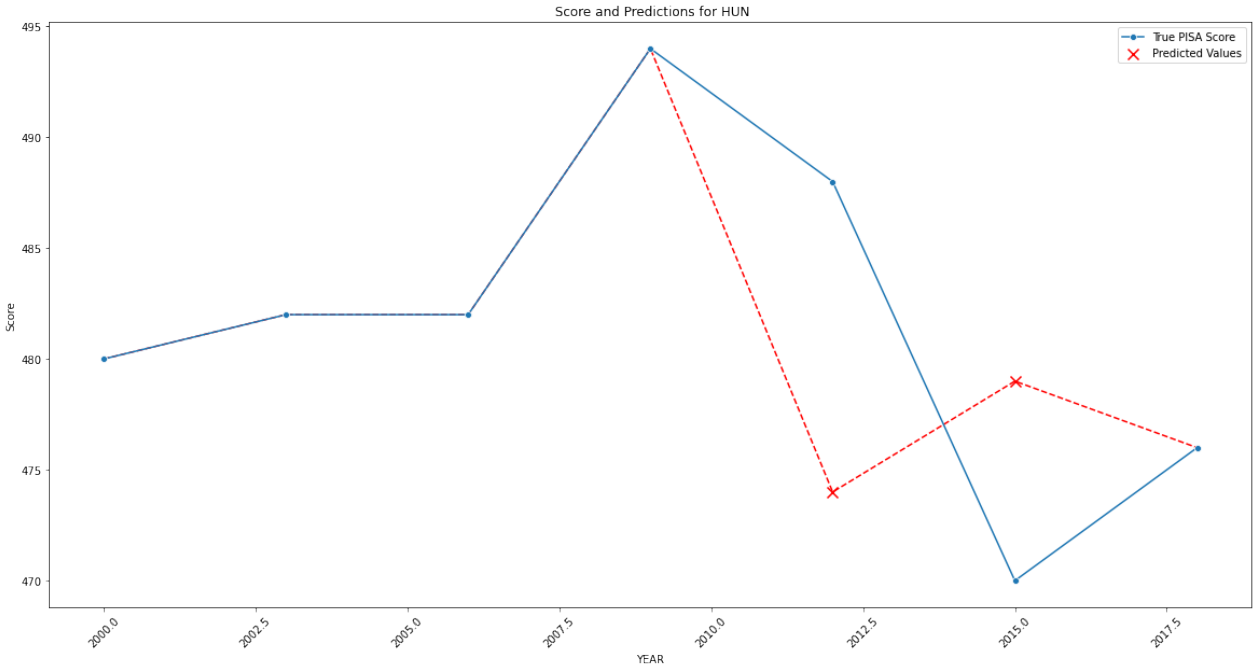

Score and Predictions for PER
True PISA Score
Predicted Values
398.0
397.8
397.6
397.4
397.2
397.0
Score
1925
1950
1975
2000
2025
2050
2075
2100
2125
YEAR

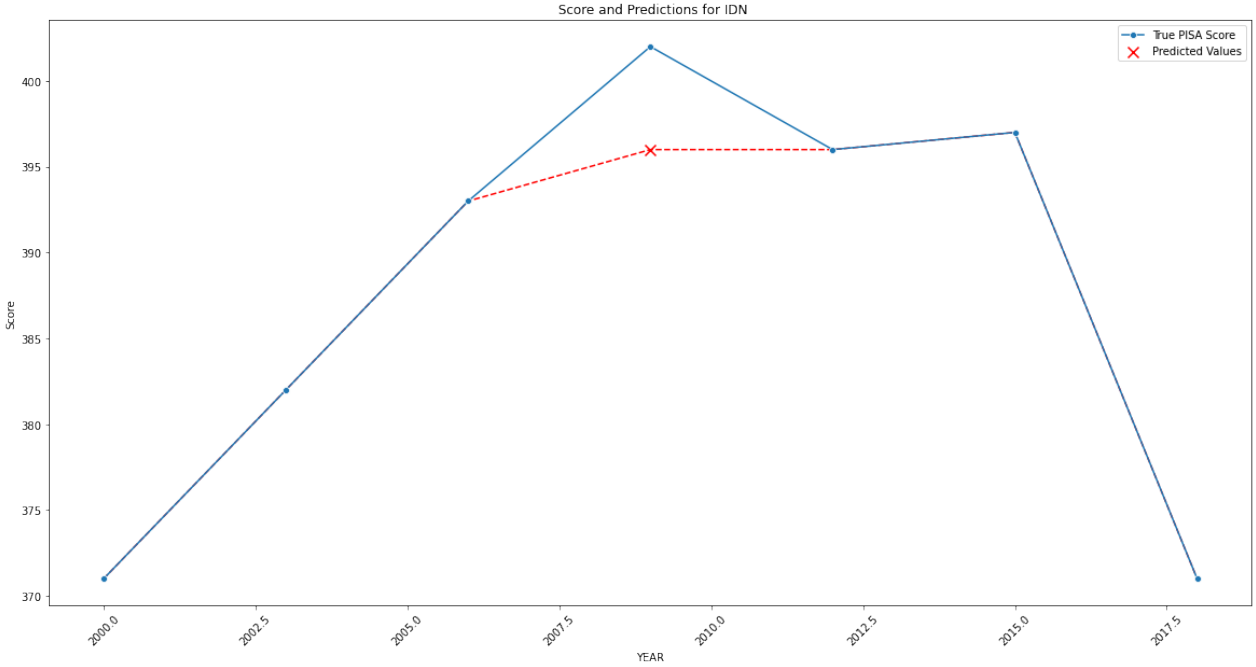

Score and Predictions for SWE
True PISA Score
Predicted Values
515
510
505
500
495
490
485
Score
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
YEAR

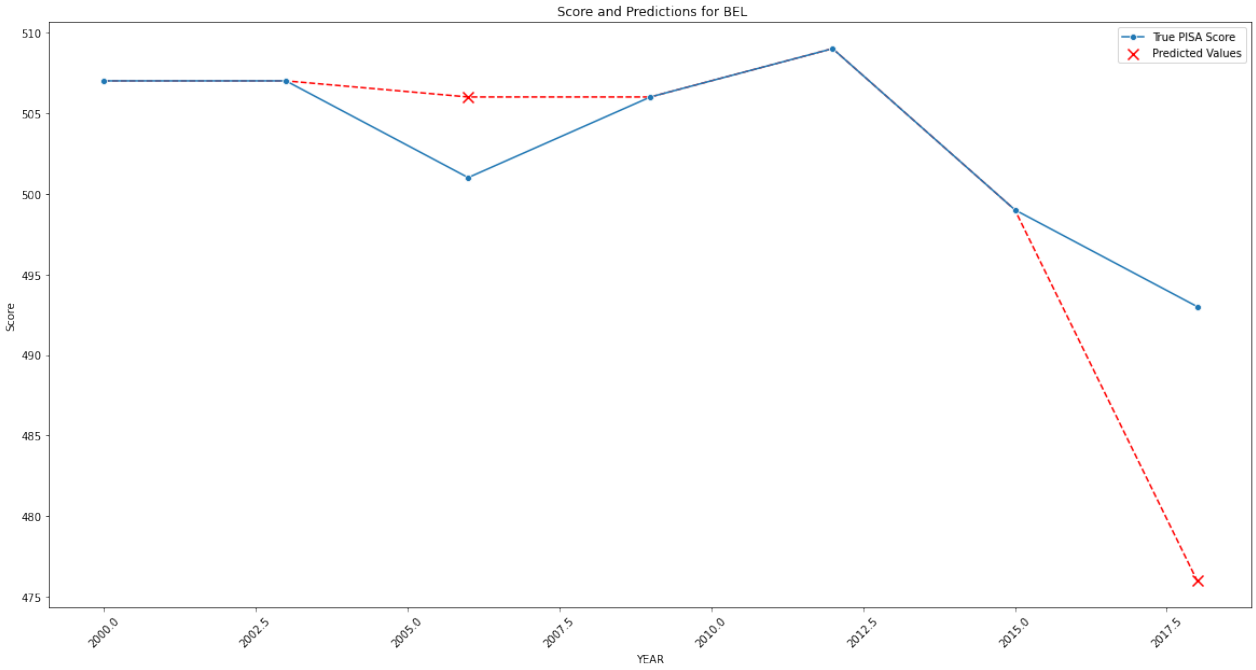

Score and Predictions for AUS
True PISA Score
Predicted Values
Score
525
520
515
510
505
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
YEAR

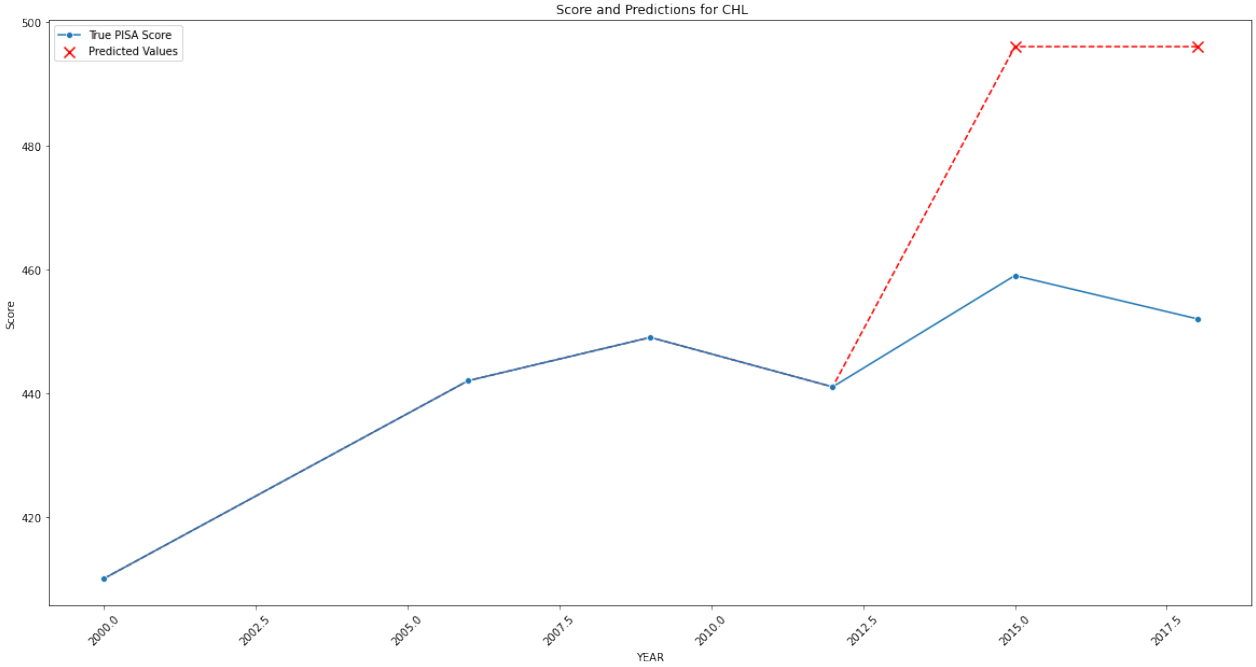

Score and Predictions for PRT
True PISA Score
Predicted Values
Score
495
490
485
480
475
470
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
YEAR

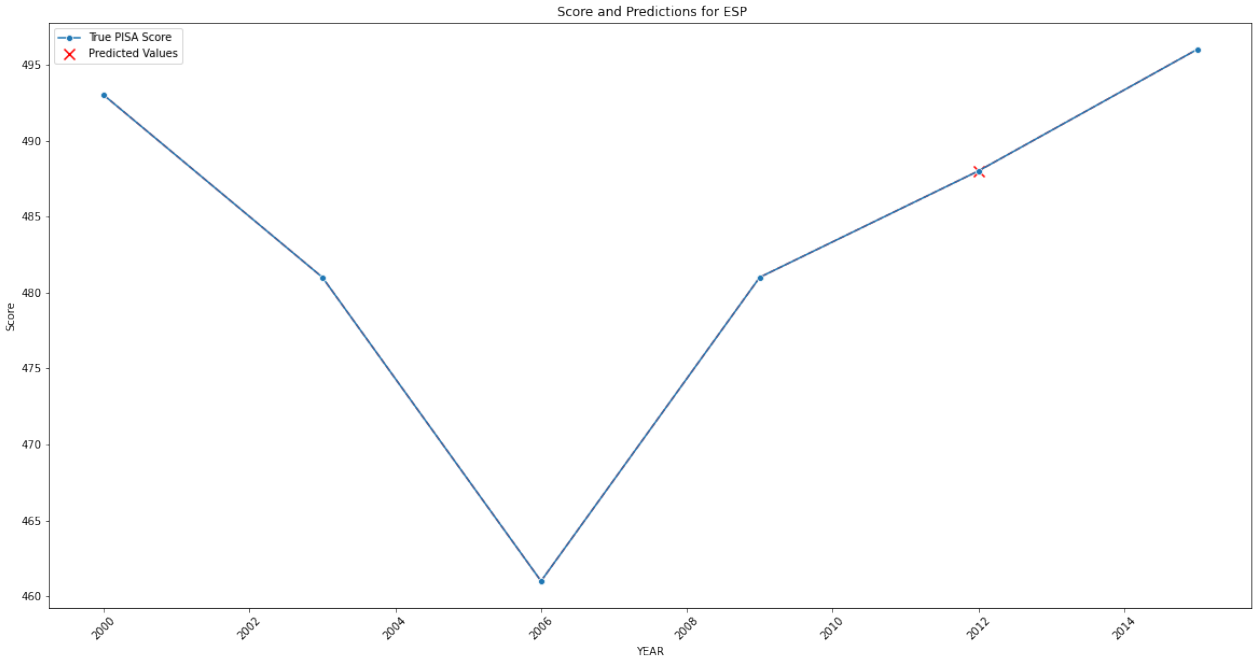

Score and Predictions for GRC
True PISA Score
Predicted Values
Score
480
475
470
465
460
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
YEAR

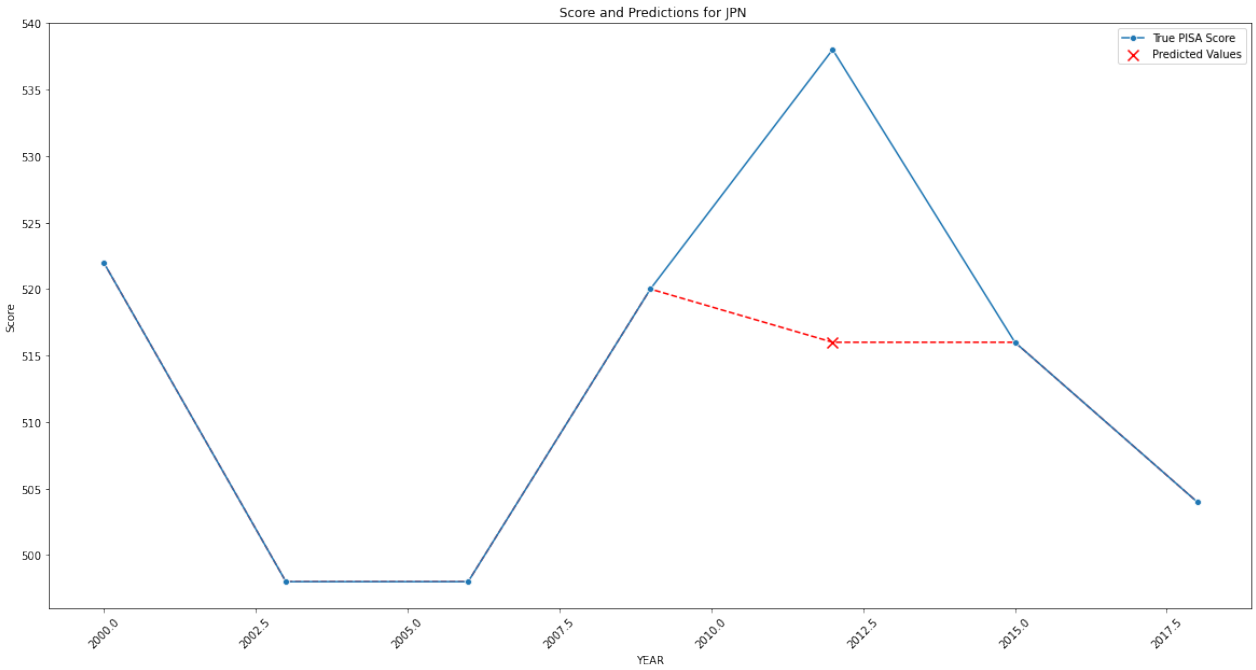

Score and Predictions for HUN
True PISA Score
Predicted Values
Score
495
490
485
480
475
470
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
YEAR

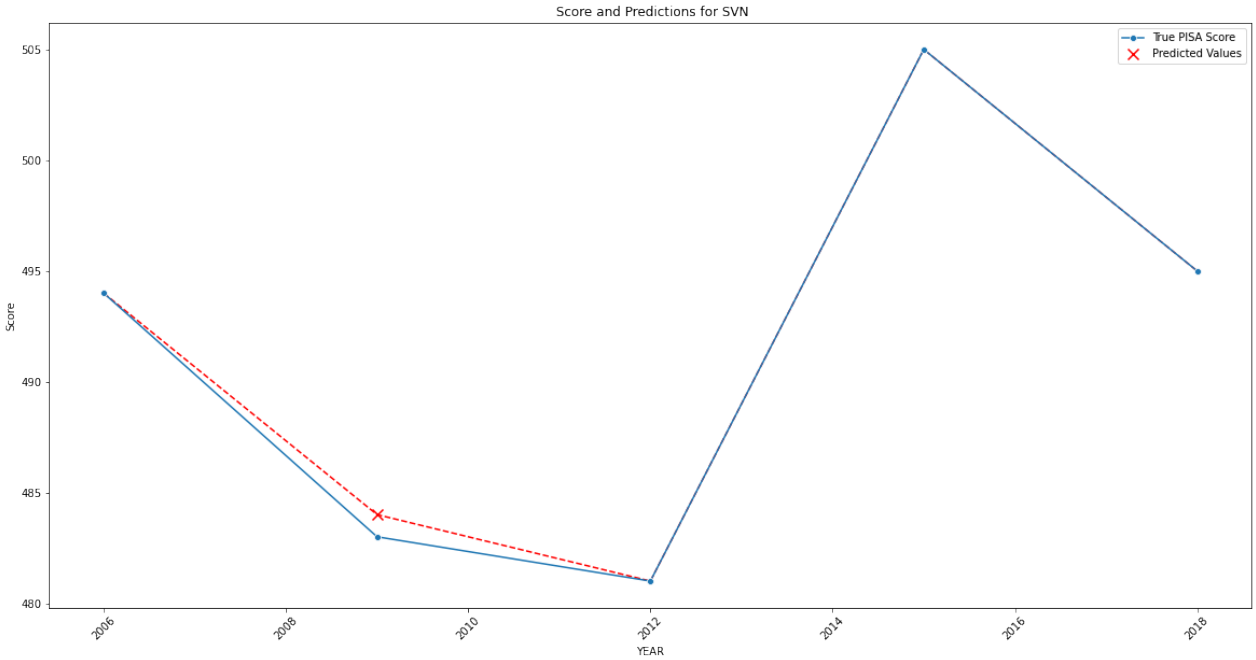

Score and Predictions for IDN
True PISA Score
Predicted Values
Score
YEAR
400
395
390
385
380
375
370
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

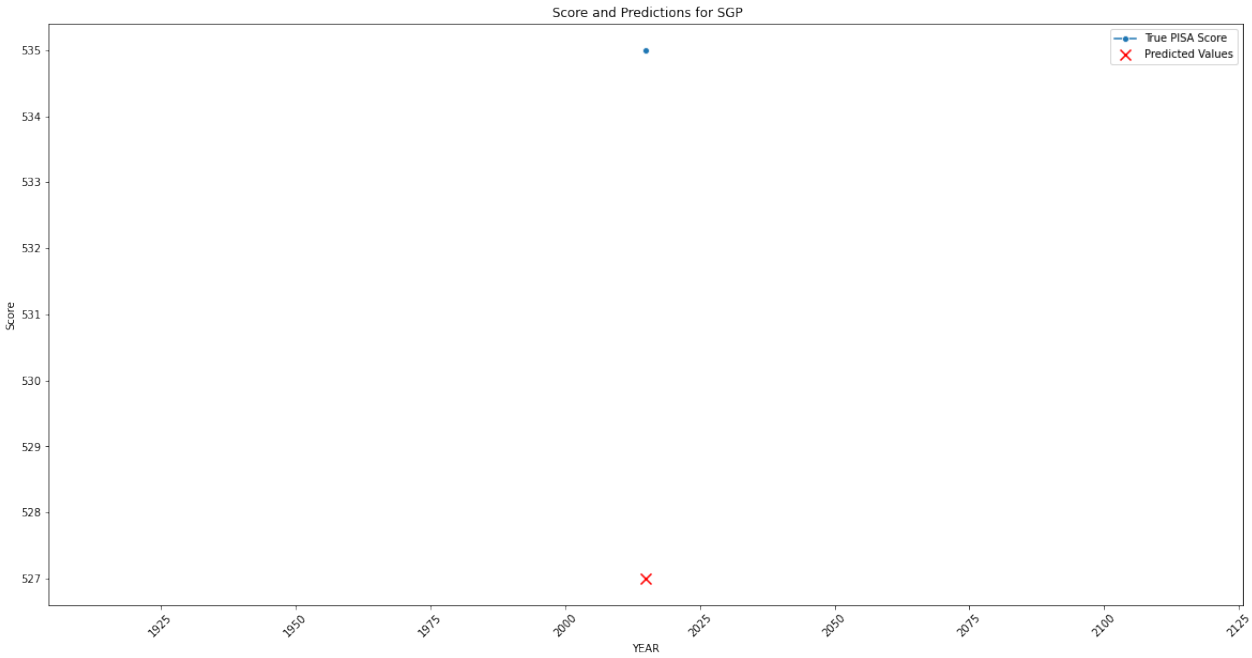

Score and Predictions for BEL
True PISA Score
Predicted Values
Score
YEAR
510
505
500
495
490
485
480
475
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

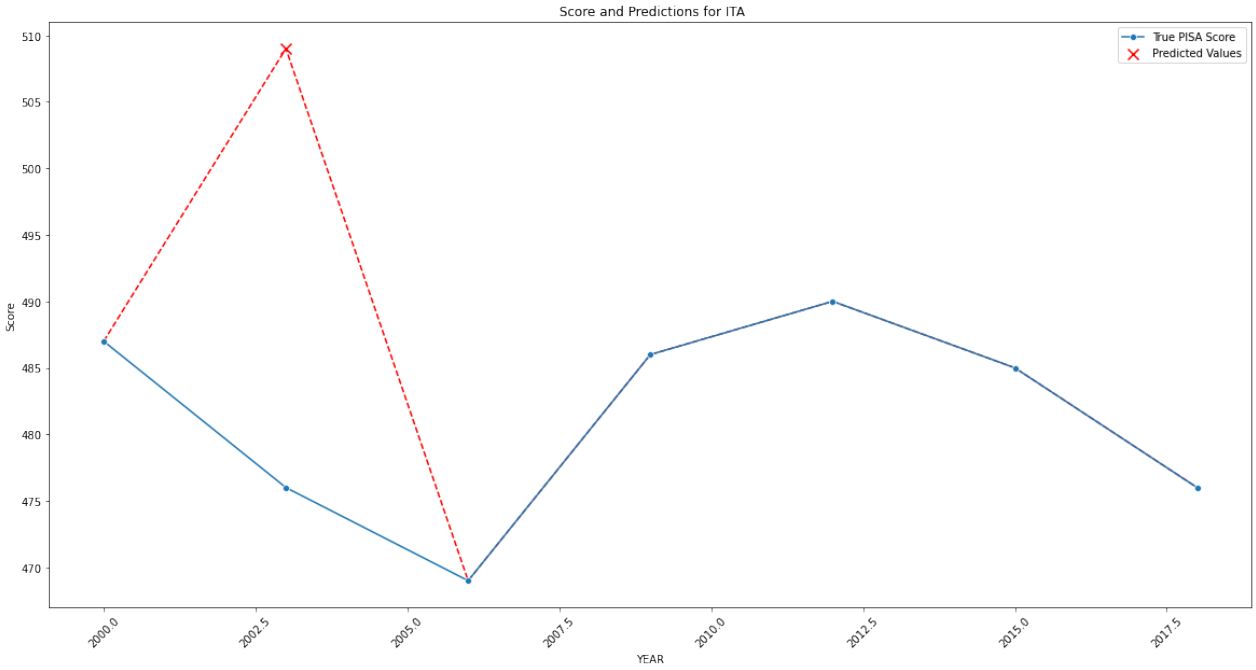

Score and Predictions for CHL
True PISA Score
Predicted Values
Score
YEAR
500
480
460
440
420
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

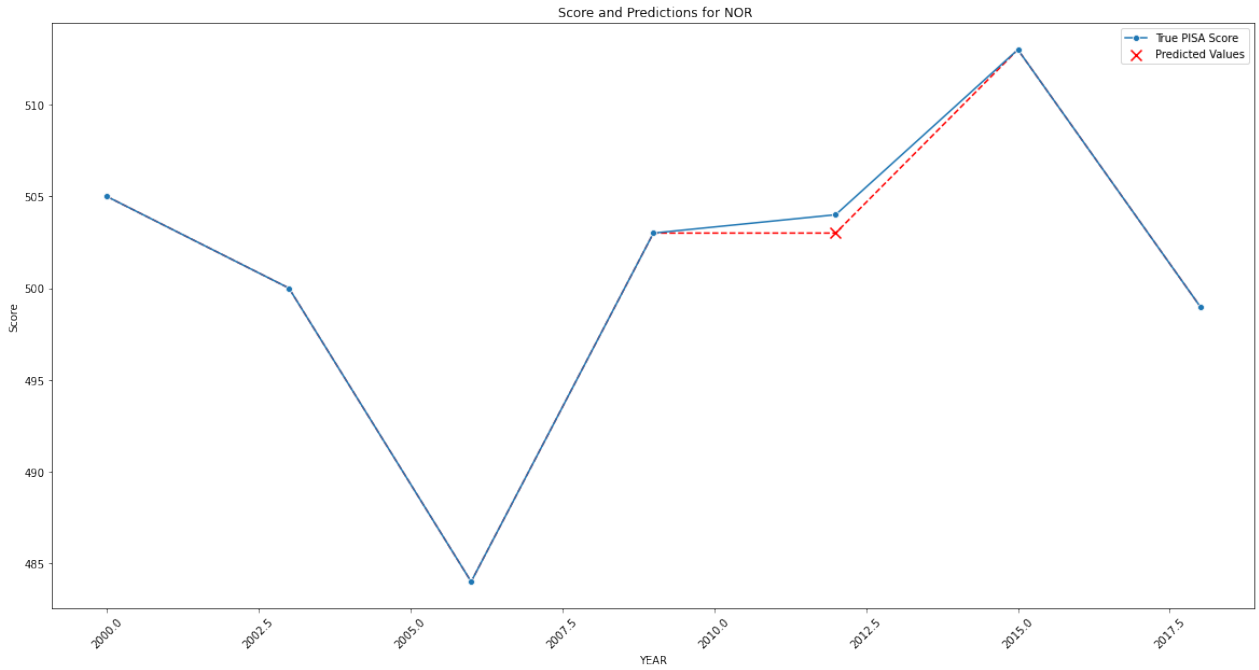

Score and Predictions for ESP
True PISA Score
Predicted Values
Score
YEAR
495
490
485
480
475
470
465
460
2000
2002
2004
2006
2008
2010
2012
2014


Score and Predictions for JPN
True PISA Score
Predicted Values
Score
YEAR
540
535
530
525
520
515
510
505
500
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

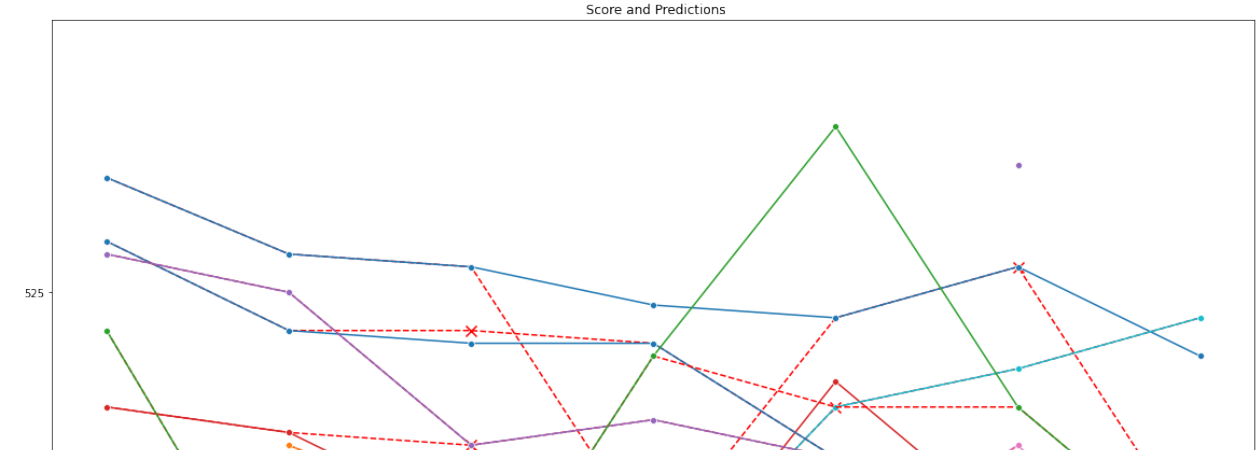

Score and Predictions for SVN
True PISA Score
Predicted Values
Score
YEAR
505
500
495
490
485
480
2006
2008
2010
2012
2014
2016
2018

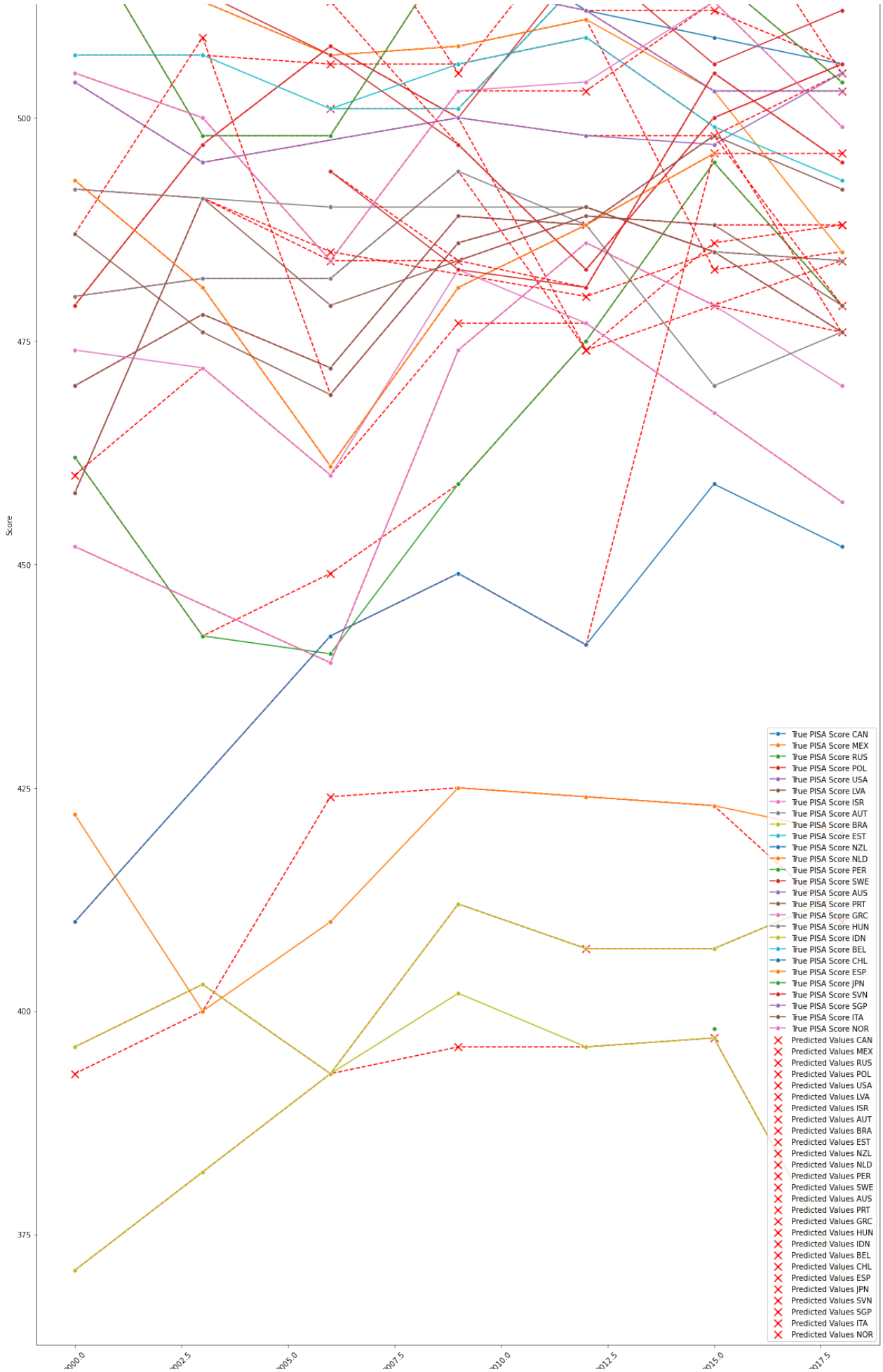

Score and Predictions for SGP
True PISA Score
Predicted Values
Score
YEAR
535
534
533
532
531
530
529
528
527
1925
1950
1975
2000
2025
2050
2075
2100
2125

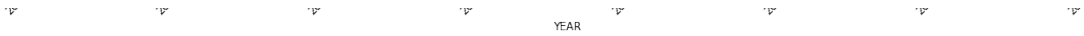

Score and Predictions for ITA
True PISA Score
Predicted Values
Score
YEAR
510
505
500
495
490
485
480
475
470
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5



In [72]:
```
###all countries in one plot - maybe an annual systematic error might be s
pottet on country basis###
plt.figure(figsize=(20,40))
for country in  estimated_countries:

    tmp = estimated_countries_data.loc[estimated_countries_data["COUNTRY"]
== country].copy()

    s = sns.lineplot(data = tmp, x="YEAR", y="new_pisa", color='r', linest
yle='--')
    p = sns.lineplot(data = tmp, x="YEAR", y="PISA-SCORE", marker='o', lab
el=f"True PISA Score {country}")
    plt.scatter(tmp["YEAR"], tmp["y_predict"], color='r', marker='x', labe
l=f"Predicted Values {country}", s=100)
    p.get_xaxis().get_major_formatter().set_useOffset(False)

plt.xticks(rotation=45)
plt.title(f" Score and Predictions")
plt.ylabel("Score")
plt.legend(loc=4)
plt.show()
plt.close()
```




Score
500
475
450
425
400
375
True PISA Score CAN
True PISA Score MEX
True PISA Score RUS
True PISA Score POL
True PISA Score USA
True PISA Score LVA
True PISA Score ISR
True PISA Score AUT
True PISA Score BRA
True PISA Score EST
True PISA Score NZL
True PISA Score NLD
True PISA Score PER
True PISA Score SWE
True PISA Score AUS
True PISA Score PRT
True PISA Score GRC
True PISA Score HUN
True PISA Score IDN
True PISA Score BEL
True PISA Score CHL
True PISA Score ESP
True PISA Score JPN
True PISA Score SVN
True PISA Score SGP
True PISA Score ITA
True PISA Score NOR
Predicted Values CAN
Predicted Values MEX
Predicted Values RUS
Predicted Values POL
Predicted Values USA
Predicted Values LVA
Predicted Values ISR
Predicted Values AUT
Predicted Values BRA
Predicted Values EST
Predicted Values NZL
Predicted Values NLD
Predicted Values PER
Predicted Values SWE
Predicted Values AUS
Predicted Values PRT
Predicted Values GRC
Predicted Values HUN
Predicted Values IDN
Predicted Values BEL
Predicted Values CHL
Predicted Values ESP
Predicted Values JPN
Predicted Values SVN
Predicted Values SGP
Predicted Values ITA
Predicted Values NOR



In den jeweiligen Skripten exportieren wir alle Plots und Tabellen. Eine Modellauf besteht zudem aus dem trainieren und testen verschiedener Modelle. Zudem haben wir am Ende jedes Laufs eine Tabelle mit den wichtigsten Daten des Laufs aus der "Results" Tabelle erstellt. In den jeweiligen Läufen haben wir auch zum Teil zusätzliche/andere Parameter mit ausgegeben (z.B. alphas und iMpurity beim Pruning, Hyperparameter, ...)

Wir haben die folgenden Modelläufe für alle Pisa Typen initiiert:

1. Ein Modell ohne jewglich Einschränkungen je non-NA Anteil und Input Daten Typ (Single_Tree_No_Restrictions_All_Input.py)
2. Ausprobieren verschiedener Hyperparameter um die Effekte zu Beobachten und zu verstehen (nur einmalig mit einem Pisa) (Single_tree_Hyperparameter_Experimentation.py)
3. Optimierung der Hyperparameter je non-NA Anteil und Input Daten Typ (Single_Tree_Optimisation.py)
4. Pruning ohne Einschränkungen auf allen Input-Daten (Pruning_no_Restrictions.py)

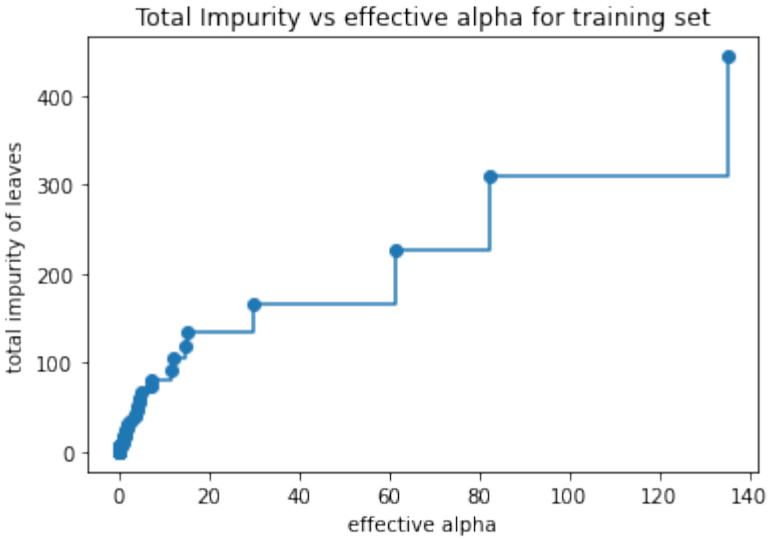
Mit ausgewählten Input-Daten haben wir folgende Modelläufe für die Analyse der Pisa Daten gestartet:

1. Pruning mit allen effizienten Alphas auf den optimalen Bäumen je non-NA Anteil und Input Daten Typ (Pruning_on_optimal_Tree.py)
2. Optimiertes Pruning auf ausgewählten Input Daten (Pruning_Optimisation.py)
3. Random Forest mit einem Beispieldatensatz zum Ausprobieren der verschiedenen Hyperparameter (Random_Forest_Parameter_Experimentation.py)
4. Random Forest ohne jegliche einschränkungen auf ausgewählten Input Daten (Random_Forest_no_restrictions.py)
5. Random Forest mit den Hyperparametern der optimierten Bäume (Random_Forest_on_optimal_Tree.py)
6. Random Forest mit Hyperparametern der optimierten Bäume und ausgewähltem Alpha für das Pruning (Random_Forest_pruned_Tree.py)

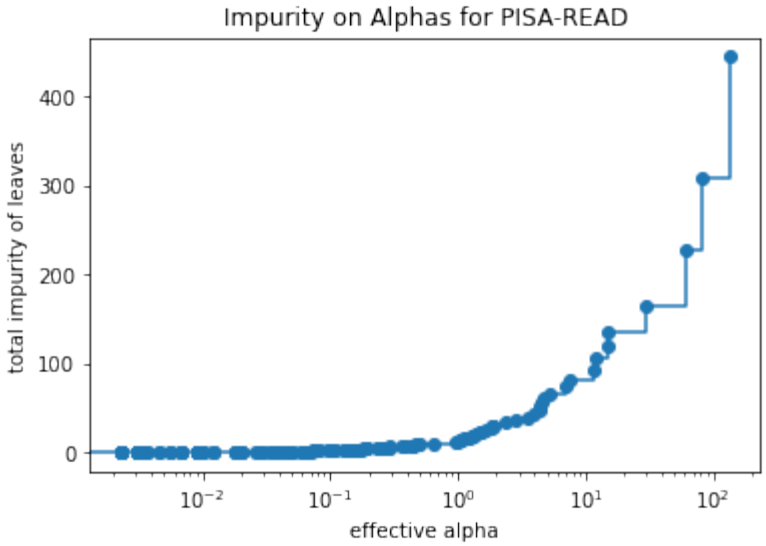
Beim Pruning wurde zudem ein vergleich über verschiedene Alphas ergänzt:

In [10]:
```
# prepare Data
feature_columns = [x for x in data_for_model.columns if x != "PISA-SCORE"
and x != "COUNTRY"]
X = data_for_model[feature_columns]
y = data_for_model["PISA-SCORE"]

# Split Data
X_train_final, X_test_final, y_train_final, y_test_final = train_test_spli
t(X, y, test_size=0.15,

  random_state=42)

# create and train model
model_tree = DecisionTreeRegressor()
path = model_tree.cost_complexity_pruning_path(X_train_final, y_train_fina
l)
```

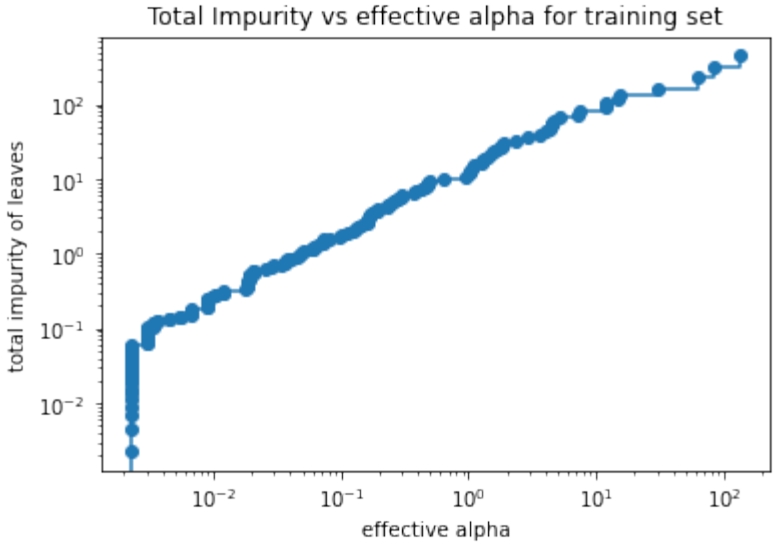
```
ccp_alphas, impurities = path.ccp_alphas, path.impurities
alphas_impurities = dict(zip(ccp_alphas, impurities))

# Plot Impurity vs. effective alphas
plt.figure()
plt.plot(ccp_alphas[:-1], impurities[:-1], marker='o', drawstyle="steps-po
st")
plt.xlabel("effective alpha")
plt.ylabel("total impurity of leaves")
plt.title("Total Impurity vs effective alpha for training set")
plt.show()
plt.close()
```



In [11]:
```
# to ease visibility in für the low alphas: logarithmic scale
# Plot Imupority vs. effective alphas
plt.figure()
plt.plot(ccp_alphas[:-1], impurities[:-1], marker='o', drawstyle="steps-po
st")
plt.xlabel("effective alpha")
plt.ylabel("total impurity of leaves")
plt.xscale("log")
plt.title(f"Impurity on Alphas for PISA-{pisa_type}")
plt.show()
plt.close()
```



In [12]:
```
# looks very exponential - add logarithmic scale for y-axis
plt.figure()
plt.plot(ccp_alphas[:-1], impurities[:-1], marker='o', drawstyle="steps-po
st")
plt.xlabel("effective alpha")
plt.ylabel("total impurity of leaves")
plt.xscale("log")
plt.yscale("log")
plt.title("Total Impurity vs effective alpha for training set")
plt.show()
plt.close()
```

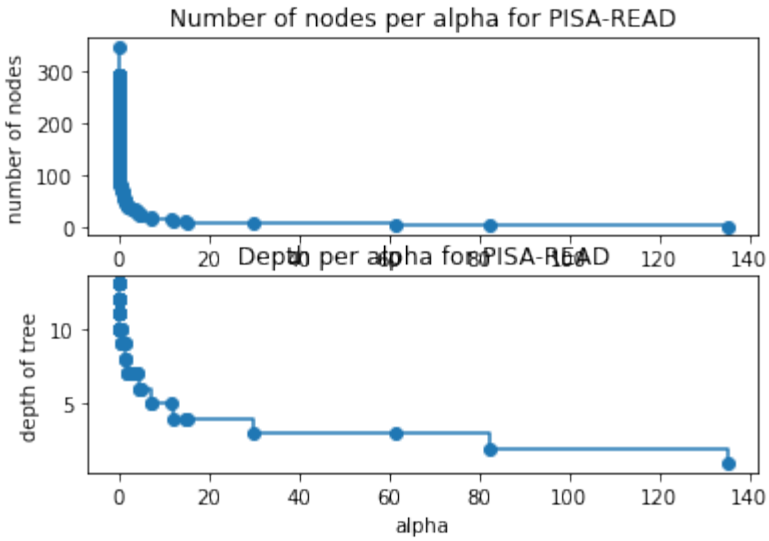


In [13]:
```
%%time
regressors = []
for ccp_alpha in ccp_alphas:
    model_tree = DecisionTreeRegressor(ccp_alpha=ccp_alpha)
    model_tree.fit(X_train_final, y_train_final)
    regressors.append(model_tree)

# when all alphas run through - Plots for different alphas
# plot showing different depths
```

```
regressors = regressors[:-1]
ccp_alphas_no_root = ccp_alphas[:-1]

node_counts = [regressor.tree_.node_count for regressor in regressors]
depth = [regressor.tree_.max_depth for regressor in regressors]
fig, ax = plt.subplots(2, 1)
ax[0].plot(ccp_alphas_no_root, node_counts, marker='o', drawstyle="steps-p
ost")
ax[0].set_xlabel("alpha")
ax[0].set_ylabel("number of nodes")
ax[0].set_title(f"Number of nodes per alpha for PISA-{pisa_type}")
ax[1].plot(ccp_alphas_no_root, depth, marker='o', drawstyle="steps-post")
ax[1].set_xlabel("alpha")
ax[1].set_ylabel("depth of tree")
ax[1].set_title(f"Depth per alpha for PISA-{pisa_type}")
fig.show()
```

```
<timed exec>:23: UserWarning: Matplotlib is currently using module://ipyke
rnel.pylab.backend_inline, which is a non-GUI backend, so cannot show the
figure.
```

```
Wall time: 19min 1s
```



In [ ]:
```
# plot showing scores per alpha
train_scores = [regressor.score(X_train_final, y_train_final) for regresso
r in regressors]
test_scores = [regressor.score(X_test_final, y_test_final) for regressor i
n regressors]

fig, ax = plt.subplots()
ax.set_xlabel("alpha")
ax.set_ylabel("R2")
ax.set_title(f"R2 per alpha for training and testing sets for PISA-{pisa_t
ype}")
ax.plot(ccp_alphas_no_root, train_scores, marker='o', label="train",
        drawstyle="steps-post")
ax.plot(ccp_alphas_no_root, test_scores, marker='o', label="test",
        drawstyle="steps-post")
ax.legend()
fig.show()
```

Beim Random Forest sind wir wie folgt vorgegegangen (ausgegebene Plots bis auf die Robustness (s.u.) gleich):

In [23]:
```
%%time

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance

# prepare Data
feature_columns = [x for x in data_for_model.columns if x != "PISA-SCORE"
and x != "COUNTRY"]
X = data_for_model[feature_columns]
y = data_for_model["PISA-SCORE"]

# Split Data
X_train_final, X_test_final, y_train_final, y_test_final = train_test_spli
t(X, y, test_size=0.15)

# create model
model_forest = RandomForestRegressor()

# train model
model_forest.fit(X_train_final, y_train_final)

# get child estimators
child_estimators = model_forest.estimators_

# evaluate model
y_predict = model_forest.predict(X_test_final)
```

```
Wall time: 19min 16s
```

Zur Analyse der verwendeten Features ziehen wir neben der Feature Importance des Regressors auch die Permutational Importance zu Rate. Diese berechnen wir einmal für die Test und einmal für die Trainingsdaten, da sich hier Diskrepanzen ergeben können.

In [24]:
```
%%time
# get feature permutation imortance on test-data
perm_import_test = permutation_importance(model_forest, X_test_final, y_te
st_final)
```

```
Wall time: 17min 48s
```

In [25]:
```
%%time
# get feature permutation imortance on training-data
perm_import_train = permutation_importance(model_forest, X_train_final, y_
train_final)
```

```
Wall time: 15min 5s
```

In [27]:
```
# save information about features
features = X.columns.to_list()
features_df = pd.DataFrame(index=features)
```

```
features_df["Feature Importance"] = model_forest.feature_importances_
features_df["TEST-Permutation Importance Mean"] = perm_import_test.importa
nces_mean
features_df["TEST-Permutation Importance SD"] = perm_import_test.importanc
es_std
features_df["TRAIN-Permutation Importance Mean"] = perm_import_train.impor
tances_mean
features_df["TRAIN-Permutation Importance SD"] = perm_import_train.importa
nces_std
features_df = features_df.sort_values("TEST-Permutation Importance Mean",
ascending=False)
features_merged = features_df.merge(column_information, how="left", left_i
ndex=True, right_index=True)
features_merged
```

Out[27]:

	Feature Importance	TEST-Permutation Importance Mean	TEST-Permutation Importance SD	TRAIN-Permutation Importance Mean	TRAIN-Permutation Importance SD	NA-Values	Va
TM.TAX.MANF.BR.ZS-Bound rate, simple mean, manufactured products (%)	0.365472	0.469140	0.139361	0.383066	0.017223	8.0	3.46
SI.DST.FRST.20-Income share held by lowest 20%	0.048131	0.045579	0.027721	0.057845	0.007095	80.0	34.63
SP.POP.SCIE.RD.P6-Researchers in R&D (per million people)	0.084778	0.030000	0.017943	0.022354	0.001663	38.0	16.45
SP.ADO.TFRT-Adolescent fertility rate (births per 1,000 women ages 15-19)	0.028753	0.010508	0.005052	0.003989	0.000363	1.0	0.43
AG.LND.ARBL.HA.PC-Arable land (hectares per person)	0.005550	0.007817	0.001523	0.002500	0.000733	43.0	18.61
...	...	...	...	...	...	...	
IOTSI4_2018 - : Domestic output and imports + Mining support service activities + Agriculture, forestry and fishing [Millions US Dollar]	0.000485	-0.000437	0.000364	0.000267	0.000042	76.0	32.90
ETCR - Market structure : []	0.000999	-0.000488	0.000177	0.000329	0.000057	100.0	43.29
GREEN_GROWTH - : Population, ages 65 and above, % total [Units Percentage]	0.006094	-0.000532	0.000765	0.000448	0.000064	3.0	1.29

ANBERD_REV4 - PPP dollars - 2015 prices : Main activity + FOOD PRODUCTS, BEVERAGES AND TOBACCO + 2015.0 [Units US Dollar]	0.000297	-0.000546	0.000457	0.000085	0.000027	88.0	38.09
FISH_AQUA - US Dollar : MOLLUSCS [Units US Dollar]	0.001005	-0.000647	0.000359	0.000251	0.000065	82.0	35.49

19652 rows × 11 columns

In [28]:
```
# store information on child estimators
child_estimator_info = pd.DataFrame(index=child_estimators)
for estimator in child_estimators:
    params = estimator.get_params()
    child_estimator_info.loc[estimator, "Parameter"] = model_run
    child_estimator_info.loc[estimator, "Percentage non-NA"] = iteration
    child_estimator_info.loc[estimator, "R2"] = estimator.score(X_test_fin
al, y_test_final)
    child_estimator_info.loc[estimator, "Random State"] = params["random_s
tate"]
    child_estimator_info.loc[estimator, "Depth"] = estimator.get_depth()
    child_estimator_info.loc[estimator, "Number of Features"] = estimator.
max_features_
    child_estimator_info.loc[estimator, "Percentage of Features"] = estima
tor.max_features
    child_estimator_info.loc[estimator, "Number Leaves"] = estimator.get_n
_leaves()
    child_estimator_info.loc[estimator, "Feature Importance"] = str(
        estimator.feature_importances_)  # return to list with json lib ->
 json.loads()
    child_estimator_info.loc[estimator, "Predictions"] = str(
        estimator.predict(X_test_final))  # return to list with json lib -
> json.loads()
child_estimator_info
```

Out[28]:

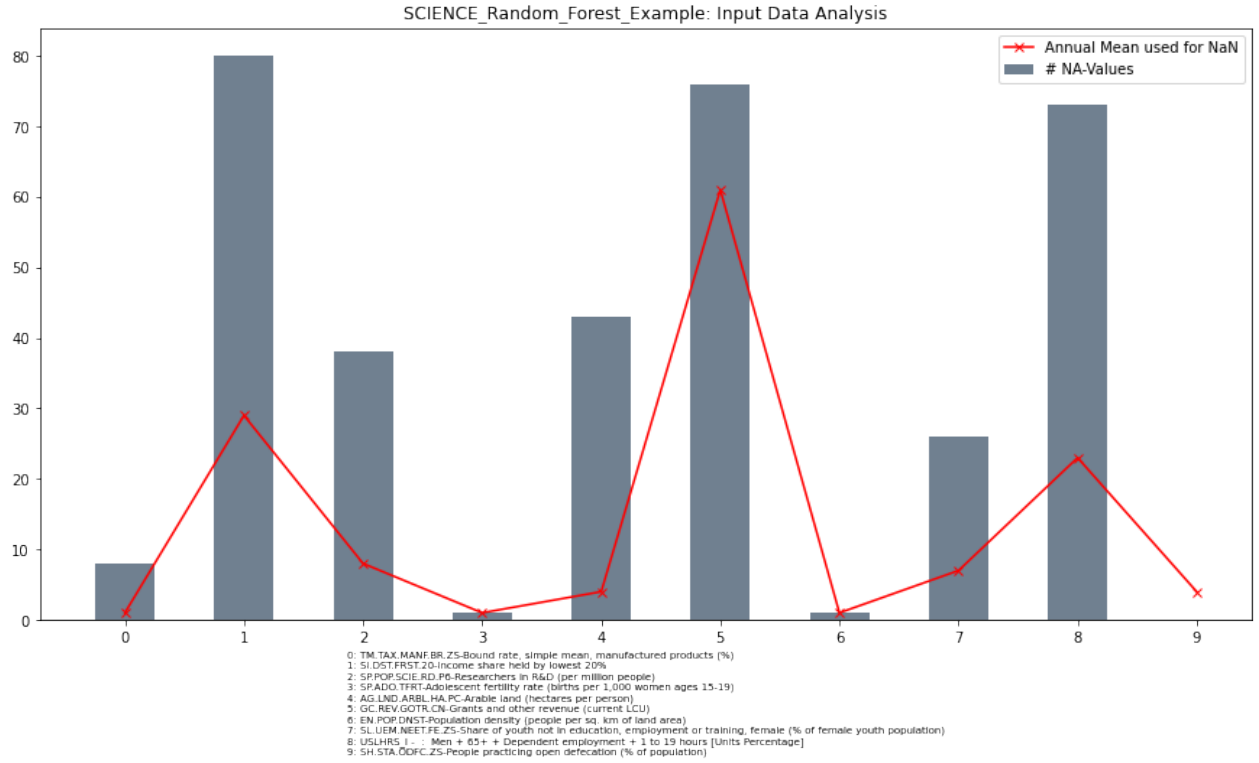
	Parameter	Percentage non-NA	R2
DecisionTreeRegressor(max_features='auto', random_state=1608637542)	Random_Forest_no_restrictions_RS42_2	65.0	0.565092
DecisionTreeRegressor(max_features='auto', random_state=1273642419)	Random_Forest_no_restrictions_RS42_2	65.0	0.902750
DecisionTreeRegressor(max_features='auto',			

random_state=1935803228)	Random_Forest_no_restrictions_RS42_2	65.0	0.751211
DecisionTreeRegressor(max_features='auto', random_state=787846414)	Random_Forest_no_restrictions_RS42_2	65.0	0.833685
DecisionTreeRegressor(max_features='auto', random_state=996406378)	Random_Forest_no_restrictions_RS42_2	65.0	0.821938
...	...	...	...
DecisionTreeRegressor(max_features='auto', random_state=893102645)	Random_Forest_no_restrictions_RS42_2	65.0	0.788737
DecisionTreeRegressor(max_features='auto', random_state=200619113)	Random_Forest_no_restrictions_RS42_2	65.0	0.807649
DecisionTreeRegressor(max_features='auto', random_state=290770691)	Random_Forest_no_restrictions_RS42_2	65.0	0.904588
DecisionTreeRegressor(max_features='auto', random_state=793943861)	Random_Forest_no_restrictions_RS42_2	65.0	0.780207
DecisionTreeRegressor(max_features='auto', random_state=134489564)	Random_Forest_no_restrictions_RS42_2	65.0	0.882148

100 rows × 10 columns

In [33]:
```
### Robustness-Check ###
plt.figure(figsize=(15, 15))
feature_names = features_merged.index.to_list()[:10]
feature_names = [f"{x}: {feature_names[x]}" for x in range(len(feature_nam
es))]
feature_names = "\n".join(feature_names)
data = features_merged[:10].copy()

plt.plot(range(len(data.index)), data["Annual Mean"], marker="x", c="r", l
abel="Annual Mean used for NaN")
```

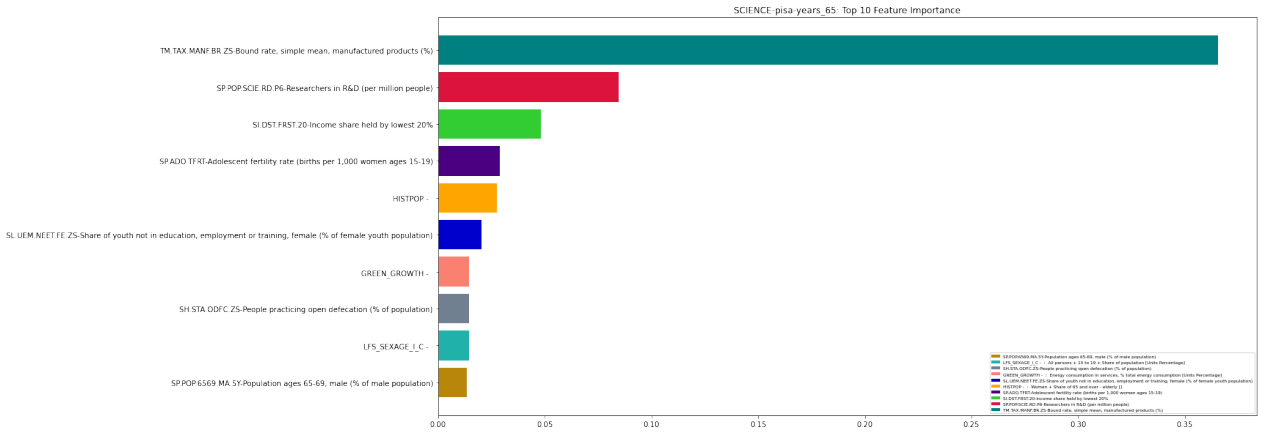
```
plt.hist(data.index, bins=range(len(data.index)), color="slategray", align
="left", rwidth=0.5,
         weights=data["NA-Values"], label="# NA-Values")
plt.legend()
plt.xticks(range(len(data.index)), range(len(data.index)));
plt.xlabel(feature_names, fontsize="x-small", ma="left")
plt.title(f"{pisa_type}_{model_run}: Input Data Analysis")
plt.gcf().subplots_adjust(bottom=0.5)
plt.show()
plt.close()
```

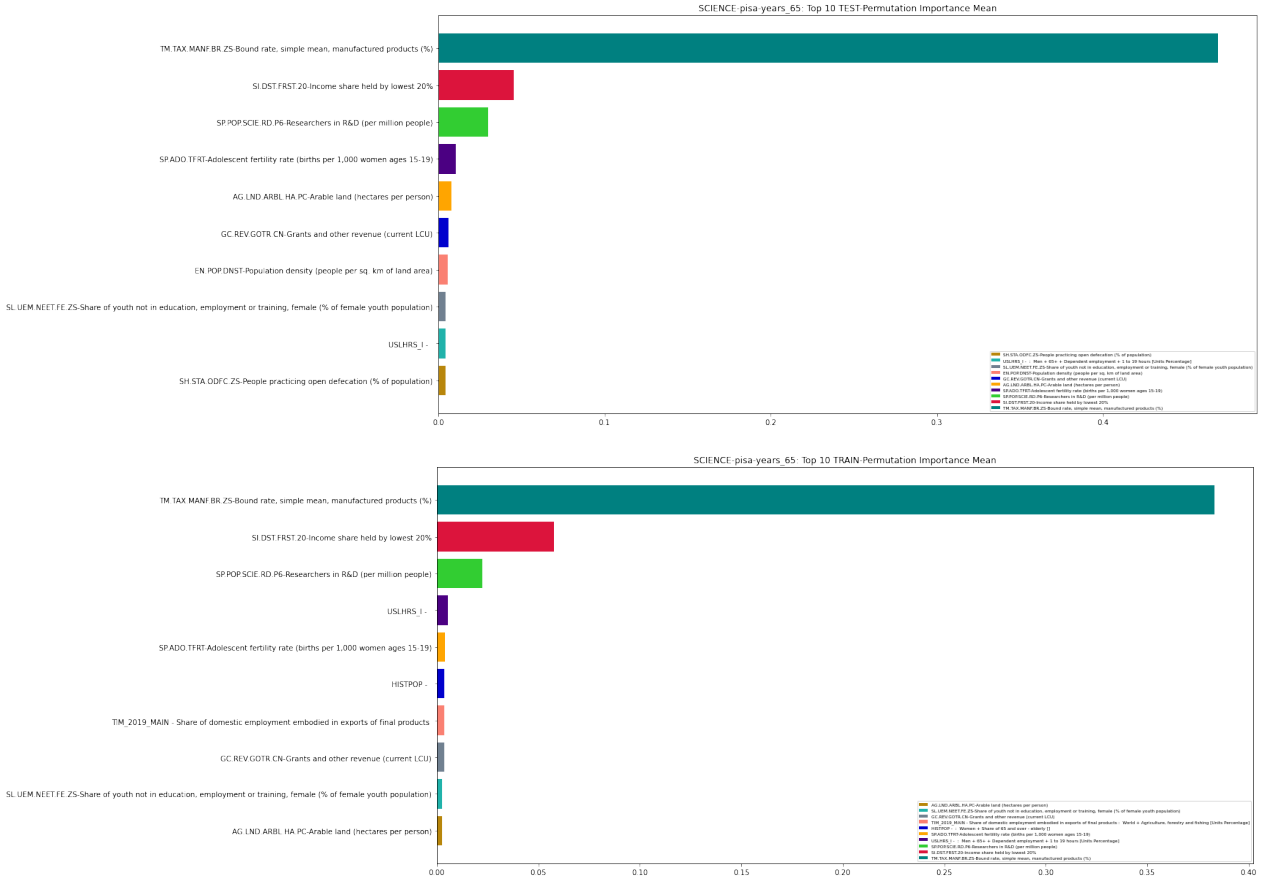


In [40]:

```
# compare the different measures of feature importance
###Feature Analysis###
Importance_Measures = ["Feature Importance", "TEST-Permutation Importance
Mean",
                       "TRAIN-Permutation Importance Mean"]
imp_df = features_df.copy()
all_measures = []
colors = ["teal", "crimson", "limegreen", "indigo", "orange", "mediumblue"
, "salmon",
          "slategray", "lightseagreen", "darkgoldenrod"]
for importance_measure in Importance_Measures:
    features_df = features_df.sort_values(importance_measure, ascending=Fa
lse)
    data = features_df[:10].copy()
    data["name"] = [x.split(":")[0] for x in data.index]
    data["labels"] = [str(x) for x in range(len(data))]
    data["color"] = colors[:len(data)]
    data = data.sort_values(importance_measure)

    all_measures.extend(data.index.to_list())

    fig = plt.figure(figsize=(20, 10))
```

```
    for f in data.index:
        plt.barh(data.loc[f, "labels"], data.loc[f, importance_measure], a
lign="center",
                 tick_label=data.loc[f, "name"], color=data.loc[f, "color"
])

    plt.title(f"{pisa_type}-{input_data}_{iteration}: Top 10 {importance_m
easure}")
    plt.legend(labels=data.index, ncol=1, bbox_transform=fig.transFigure,
loc="lower right",
               fontsize='xx-small')
    plt.yticks(ticks=np.arange(len(data)), labels=data["name"])
    plt.show()
    plt.close()

all_measures = set(all_measures)
plot_df = features_df.loc[all_measures, :].copy()
names = [f"{x}: {plot_df.index.to_list()[x]}" for x in range(len(plot_df.i
ndex))]
names_overview = "\n".join(names)

fig = plt.figure(figsize=(10, 10))
plt.title(f"{pisa_type}-{model_run}: Measures of Feature Importance")
plt.scatter(x=range(len(names)), y=plot_df["Feature Importance"], label="F
eature Importance")
plt.scatter(x=range(len(names)), y=plot_df["TEST-Permutation Importance Me
an"], label="TEST Permutation Importance")
plt.scatter(x=range(len(names)), y=plot_df["TRAIN-Permutation Importance M
ean"],
            label="TRAIN Permutation Importance")
plt.xticks(ticks=range(len(names)), fontsize="small")
plt.xlabel(names_overview, fontsize="x-small", ma="left")
plt.legend(loc="upper left", fontsize="small")
plt.gcf().subplots_adjust(bottom=0.3)
fig.show()
plt.close()
```






```
<ipython-input-40-271649afdc78>:50: UserWarning: Matplotlib is currently u
sing module://ipykernel.pylab.backend_inline, which is a non-GUI backend,
so cannot show the figure.
  fig.show()
```

Da wir nach den Modelläufen viele Ergebnisse direkt in Excel sehen oder die Diagramme betrachten können, lässt sich vieles aus den Outputs der Skripte bereits ablesen. Auf die, unser Meinung nach, wichtigsten Punkte möchten wir allerdings auch hier eingehen und diese hier nochmal veranschaulichen.

Nachdem wir die ersten paar Modelldurchläufe verglichen und analysiert haben, haben wir etwas unerwartetes festgestellt: Die Scores vom auf Hyperparameter optimierten Baum und die vom geprunten Baum mit den selber Hyperparametern und alpha = 0 waren nicht gleich, obwohl die selben Testdaten verwendet wurden. Dies war bei allen Pisas und allen Inputdaten der Fall. Beispiel Math, aggregated years

In [23]:
```
# Paths to Results and Tree-Information
path_results_opt_single_tree = Path(path_results / "aggregated-years_Optim
al_Single_Tree")
path_results_pruning_on_opt_tree = Path(path_results / "aggregated-years_P
runing_on_opt_Tree")

# load overview Tables
overview_opt_single_tree = pd.read_csv(path_results_opt_single_tree / "MAT
H_Overview_aggregated-years_Optimal_Single_Tree.csv", index_col="Unnamed:
0")
overview_pruning = pd.read_csv(path_results_pruning_on_opt_tree / "MATH_Ov
erview_aggregated-years_Pruning_on_Opt_Tree.csv", index_col=["Unnamed: 0"]
```

```
)

overview_pruning
```

Out[23]:

	Percentage non-NA	Alpha	Total Features	Features Tree	RMSE	R2	max_Depth
(70, 0.0)	70	0.000000	15417	15	12.829965	0.922118	5.0
(70, 0.0025510204081632647)	70	0.002551	15417	15	11.350053	0.939049	5.0
(70, 0.003401360543722724)	70	0.003401	15417	15	12.238574	0.929132	5.0
(70, 0.003401360544168191)	70	0.003401	15417	15	12.004568	0.931816	5.0
(70, 0.003401360544613657)	70	0.003401	15417	15	12.154599	0.930101	5.0
...	...	...	...	...	...	...	..
(100, 50.22479059806688)	100	50.224791	27	3	23.653085	0.735295	3.0
(100, 75.49534041258346)	100	75.495340	27	2	24.426439	0.717702	2.0
(100, 87.36870537689093)	100	87.368705	27	1	24.309530	0.720398	2.0
(100, 204.94139098898515)	100	204.941391	27	1	22.836915	0.753247	1.0
(100, 822.6394111437039)	100	822.639411	27	0	46.821698	-0.037246	0.0

743 rows × 11 columns

In [24]:
```
overview_opt_single_tree.loc[70:]
```

Out[24]:

	Percentage non-NA	Total Features	Features Tree	RMSE	R2	Percentage_NA_mean	r
70	Aggregation_Optimal_Single_Tree	15417	15	12.268875	0.928781	0.541126	
75	Aggregation_Optimal_Single_Tree	12380	20	11.633440	0.935967	0.649351	
80	Aggregation_Optimal_Single_Tree	10547	22	14.038185	0.906759	0.649351	
85	Aggregation_Optimal_Single_Tree	6257	22	13.376286	0.915344	0.606061	
90	Aggregation_Optimal_Single_Tree	4545	46	13.744280	0.910622	0.511610	
95	Aggregation_Optimal_Single_Tree	2455	28	13.491277	0.913882	1.028139	
100	Aggregation_Optimal_Single_Tree	27	19	26.835800	0.659265	NaN	

In [36]:
```
overview_pruning_alpha_zero = overview_pruning.loc[overview_pruning["Alpha
"] == 0]
overview_pruning_alpha_zero
```

Out[36]:

	Percentage non-NA	Alpha	Total Features	Features Tree	RMSE	R2	max_Depth	Nodes_Tree	Impuritie
(70, 0.0)	70	0.0	15417	15	12.829965	0.922118	5.0	31.0	0.
(75, 0.0)	75	0.0	12380	20	12.840692	0.921988	5.0	41.0	0.
(80, 0.0)	80	0.0	10547	22	12.552669	0.925448	6.0	45.0	0.
(85, 0.0)	85	0.0	6257	22	13.082955	0.919016	5.0	45.0	0.
(90, 0.0)	90	0.0	4545	46	11.888690	0.933126	8.0	97.0	0.
(95, 0.0)	95	0.0	2455	28	14.942279	0.894362	9.0	57.0	0.
(100, 0.0)	100	0.0	27	22	25.190984	0.699754	9.0	87.0	0.

Obwohl die gleichen Parameter verwendet wurden und der Baum sich nach Algorithmus so baut, dass er immer das Feature auswählt, das den MSE am meisten reduziert, bauen sich verschiedene Bäume. Ein Blick in die Informationen in einen der Bäume zeigt:

In [152]:
```
path_tree_opt_single_tree = Path(path_tree / "aggregated-years_Optimal_Sin
gle_Tree")
path_tree_pruning_on_opt_tree = Path(path_tree / "aggregated-years_Pruning
_on_opt_Tree")

# load overview Tables
features_opt_single_tree = pd.read_csv(path_results_opt_single_tree / "MAT
H-75_Feature_Analysis_aggregated-years_Optimal_Single_Tree.csv", index_col
="Feature")
features_pruning = pd.read_csv(path_results_pruning_on_opt_tree / "MATH-75
_Feature_Analysis_Alpha_0.0_aggregated-years_Pruning_on_Opt_Tree.csv", ind
ex_col="Feature")
tree_opt_single_tree = pd.read_csv(path_tree_opt_single_tree / "MATH-75_ag
gregated-years_Optimal_Single_Tree.csv")
tree_pruning = pd.read_csv(path_tree_pruning_on_opt_tree / "MATH-75_Alpha_
0.0_aggregated-years_Pruning_on_Opt_Tree.csv")
```

In [131]:
```
features_opt_not_prun = [x for x in features_opt_single_tree.index if not(
x in features_pruning.index)]
features_pruning_not_opt = [x for x in features_pruning.index if not(x in
features_opt_single_tree.index)]
```

In [153]:
```
tree_opt_single_tree_different = tree_opt_single_tree.loc[tree_opt_single_
tree["Node"].isin(features opt not prun)]
```

```
tree_opt_single_tree_different
```

Out[153]:

	Node	Threshold	Samples	Value
1	AWCOMP - Increase in net income after an incre...	0.872253	27	403.592593
2	AEI_OTHER - Energy industries : [Thousands T...	181709.820300	13	381.923077
3	RS_GBL - Tax revenue as % of total taxation : ...	9.905416	11	386.272727
7	PTRUB - : Couple with 2 children - partner's...	63.333334	14	423.714286
8	HEALTH_STAT - Years lost, /100 000 females, ag...	69.391666	11	416.818182
9	SNA_TABLE1 - Current prices, constant exchange...	11247.357420	7	421.000000
15	NRR - : Couple with 2 children - partner's e...	45.583334	69	485.710145
16	PTRUB - : Single person with 2 children + 24...	16.005000	11	456.000000
20	NRR - : Couple with 2 children - partner's e...	67.750000	11	513.454545
28	ERTR - : Tax revenue, % of total environment...	79.530495	9	530.222222
31	METR - : Couple without children - partner i...	43.916666	68	503.617647
34	HSL - Life expectancy at birth : Gap between ...	77.383335	23	531.391304
35	TENURE_FREQ - : Men + 15 to 64 + Dependent e...	5.529836	8	545.000000
38	FAMILY - Crude divorce rate (divorces per 1000...	1.008333	15	524.133333

In [154]:
```
tree_pruning_different = tree_pruning.loc[tree_pruning["Node"].isin(features_pruning_not_opt)]
tree_pruning_different
```

Out[154]:

	Node	Threshold	Samples	Value
1	FTPTC_I - : Women + 55 to 59 + Dependent emp...	6.945693e+01	27	403.592593
2	TEMP_I - : Men + 55 to 64 + Dependent employ...	1.828583e+01	14	423.714286
4	SP.POP.TOTL.FE.ZS-Population, female (% of tot...	4.708482e+09	11	416.818182
6	TM.VAL.OTHR.ZS.WT-Computer, communications and...	2.904395e+09	7	421.000000
9	HEALTH_PROC - Days : Single spontaneous deliv...	3.233290e+00	13	381.923077
10	GREEN_GROWTH - : Welfare costs of premature ...	1.002912e-01	11	386.272727
15	NRR - : Couple with 2 children - partner's e...	6.083333e+01	69	485.710145
16	METR - : Single person with 2 children + Fro...	4.266667e+01	11	456.000000
20	SHA - Share of financing scheme : Voluntary s...	4.642083e+00	11	513.454545
28	SHA - Per capita, current prices, current PPPs...	7.439866e+01	9	530.222222
31	METR - : Couple without children - partner i...	4.391667e+01	68	503.617647
34	HEALTH_STAT - Deaths per 100 000 population (s...	1.066667e+00	23	531.391304
35	HEALTH_STAT - Deaths per 100 000 males (crude ...	1.071667e+01	15	524.133333

38	TENURE_FREQ - : All persons + 15 to 64 + Dep...	5.720939e+00	8	545.000000

In [150]:
```
# Take a look at the splits
splits = tree_opt_single_tree_different.merge(tree_pruning_different, how="left", left_on="Value", right_on="Value")
splits = splits.drop(["Threshold_x", "Threshold_y"], axis=1)
splits = splits.set_index("Value")
splits
```

Out[150]:

Value	Node_x	Samples_x	Node_y	Samples_y
403.592593	AWCOMP - Increase in net income after an incre...	27	FTPTC_I - : Women + 55 to 59 + Dependent emp...	27
381.923077	AEI_OTHER - Energy industries : [Thousands T...	13	HEALTH_PROC - Days : Single spontaneous deliv...	13
386.272727	RS_GBL - Tax revenue as % of total taxation : ...	11	GREEN_GROWTH - : Welfare costs of premature ...	11
423.714286	PTRUB - : Couple with 2 children - partner's...	14	TEMP_I - : Men + 55 to 64 + Dependent employ...	14
416.818182	HEALTH_STAT - Years lost, /100 000 females, ag...	11	SP.POP.TOTL.FE.ZS-Population, female (% of tot...	11
421.000000	SNA_TABLE1 - Current prices, constant exchange...	7	TM.VAL.OTHR.ZS.WT-Computer, communications and...	7
485.710145	NRR - : Couple with 2 children - partner's e...	69	NRR - : Couple with 2 children - partner's e...	69
456.000000	PTRUB - : Single person with 2 children + 24...	11	METR - : Single person with 2 children + Fro...	11
513.454545	NRR - : Couple with 2 children - partner's e...	11	SHA - Share of financing scheme : Voluntary s...	11
530.222222	ERTR - : Tax revenue, % of total environment...	9	SHA - Per capita, current prices, current PPPs...	9
503.617647	METR - : Couple without children - partner i...	68	METR - : Couple without children - partner i...	68
531.391304	HSL - Life expectancy at birth : Gap between ...	23	HEALTH_STAT - Deaths per 100 000 population (s...	23
545.000000	TENURE_FREQ - : Men + 15 to 64 + Dependent e...	8	TENURE_FREQ - : All persons + 15 to 64 + Dep...	8
524.133333	FAMILY - Crude divorce rate (divorces per 1000...	15	HEALTH_STAT - Deaths per 100 000 males (crude ...	15

Bei einem Blick in den Baum sieht man, dass die Features and den gleichen stellen sind, den MSE gleich viel verbessern. Anhand der oben dargestellten Tabelle sieht man, dass die Features den exakt gleichen Split machen. Das Problem ist, dass wir zu wenig Daten und zu viele Features haben und es deshalb mehrere Möglichkeiten für den Best-Split beim Trainieren gibt. Der Baum nimmt

einen zufällig, der auf den Test-Daten aber unterschiedlich gut funktionieren (Test/Train-Split ist immer gleich).

Für uns bedeutet das leider, dass unsere bisherigen Modelle keinerlei Vergleichbarkeit haben. Das Setzen eines Random States im Regressor löst das Problem der Vergleichbarkeit (bei der Möglichkeit gleicher Splits wird immer der selbe Feature verwendet), allerdings ist es vollkommen willkürlich welches der Features ausgewählt wird. Wir möchten darauf hinweisen, dass dieses Vorgehen das Problem kaschiert und die durch die Bäume erreichten Scores beinahe vollkommen wilkürlich sind.

Allgemein gilt es anzumerken, dass durch diese extrem kleinen Menge an Daten, die wir vorhersagen wollen, "zufällige" Korrelationen sehr wahrscheinlich sind. Bei einer größeren Menge an Datenpunkten des Targets sind solche zufälligen "Korrelationen" deutlich unwahrscheinlicher. Unsere große Menge an Features unterstützt dieses Phänomen natürlich.

Um zumindest die Modelläufe miteinander zu vergleichen, wenden wir nun dennoch einen Random State im Regressor an und führen die Modelläufe nocheinmal aus. Die zuvor vorgestellten Skripte werden lediglich um das setzten eines Random States angebasst und nochmal auf dem Server ausgeführt.

Wie schon zuvor sticht auch hier in Auge, dass die optimierten Modelle bei einem non-NA Anteil von 100% vergleichbar gute R^2 und geringe RMSE erzielt werden. Aus diesem Grund möchten wir ein Augenmerk auf die zur Verfügung stehenden Features legen. Hierfür zeigen wir hier die Features für MATH, 100% aggregated-years:

In [156]:
```
path_results_opt_single_tree = Path(path_results / "aggregated-years_Optimal_Single_Tree")
results_overview = pd.read_csv(path_results_opt_single_tree / "MATH-100_Results_aggregated-years_Optimal_Single_Tree_RS42_2.csv", index_col="Unnamed: 0")
results_overview
```

Out[156]:

	Modelrun	COUNTRY	PISA-SCORE	y_predict
aggregated-years_Optimal_Single_Tree_RS42_2	aggregated-years_Optimal_Single_Tree_RS42_2	NaN	NaN	NaN
TUR-2003	aggregated-years_Optimal_Single_Tree_RS42_2	TUR	423.0	434.250000
ESP-2018	aggregated-years_Optimal_Single_Tree_RS42_2	ESP	481.0	498.934783
AUT-2015	aggregated-years_Optimal_Single_Tree_RS42_2	AUT	497.0	498.934783
NOR-2006	aggregated-years_Optimal_Single_Tree_RS42_2	NOR	490.0	498.000000
BEL-2015	aggregated-years_Optimal_Single_Tree_RS42_2	BEL	507.0	510.666667

SGP-2015	aggregated-years_Optimal_Single_Tree_RS42_2	SGP	564.0	513.333333
CAN-2009	aggregated-years_Optimal_Single_Tree_RS42_2	CAN	527.0	521.571429
RUS-2009	aggregated-years_Optimal_Single_Tree_RS42_2	RUS	468.0	480.625000
LVA-2015	aggregated-years_Optimal_Single_Tree_RS42_2	LVA	482.0	480.125000
ITA-2006	aggregated-years_Optimal_Single_Tree_RS42_2	ITA	462.0	515.888889
SWE-2012	aggregated-years_Optimal_Single_Tree_RS42_2	SWE	478.0	498.934783
BRA-2009	aggregated-years_Optimal_Single_Tree_RS42_2	BRA	386.0	408.000000
HUN-2003	aggregated-years_Optimal_Single_Tree_RS42_2	HUN	490.0	515.888889
NZL-2012	aggregated-years_Optimal_Single_Tree_RS42_2	NZL	500.0	498.934783
NZL-2003	aggregated-years_Optimal_Single_Tree_RS42_2	NZL	523.0	515.888889
KOR-2003	aggregated-years_Optimal_Single_Tree_RS42_2	KOR	542.0	515.888889
LUX-2009	aggregated-years_Optimal_Single_Tree_RS42_2	LUX	489.0	497.800000
CHE-2006	aggregated-years_Optimal_Single_Tree_RS42_2	CHE	530.0	532.166667
USA-2006	aggregated-years_Optimal_Single_Tree_RS42_2	USA	474.0	480.625000
POL-2012	aggregated-years_Optimal_Single_Tree_RS42_2	POL	518.0	498.934783
IRL-2003	aggregated-years_Optimal_Single_Tree_RS42_2	IRL	503.0	515.888889
MEX-2018	aggregated-years_Optimal_Single_Tree_RS42_2	MEX	409.0	408.000000
SVK-2015	aggregated-years_Optimal_Single_Tree_RS42_2	SVK	475.0	480.125000
PER-2015	aggregated-years_Optimal_Single_Tree_RS42_2	PER	387.0	386.500000
ITA-2003	aggregated-years_Optimal_Single_Tree_RS42_2	ITA	466.0	515.888889
BRA-2006	aggregated-years_Optimal_Single_Tree_RS42_2	BRA	370.0	408.000000
LUX-2012	aggregated-years_Optimal_Single_Tree_RS42_2	LUX	490.0	497.800000

HUN-2018	aggregated-years_Optimal_Single_Tree_RS42_2	HUN	481.0	498.934783
IDN-2009	aggregated-years_Optimal_Single_Tree_RS42_2	IDN	371.0	373.400000
GBR-2006	aggregated-years_Optimal_Single_Tree_RS42_2	GBR	495.0	515.888889
RUS-2015	aggregated-years_Optimal_Single_Tree_RS42_2	RUS	494.0	480.625000
DNK-2018	aggregated-years_Optimal_Single_Tree_RS42_2	DNK	509.0	518.100000
ISL-2015	aggregated-years_Optimal_Single_Tree_RS42_2	ISL	488.0	503.200000
CZE-2009	aggregated-years_Optimal_Single_Tree_RS42_2	CZE	493.0	498.934783
BEL-2018	aggregated-years_Optimal_Single_Tree_RS42_2	BEL	508.0	518.100000

In [168]:
```
features = pd.read_csv(path_results_opt_single_tree / "MATH-100_Feature_Analysis_aggregated-years_Optimal_Single_Tree_RS42_2.csv")
features
```

Out[168]:

	Feature	Samples	NA-Values	NA-Values [%]	Inpolated	Inpolated [%]	Annual Mean	Annual Mean [%]	Robustness - Samples affected by NA [Samples]	Robustr - Samp affecte An M [Samp
0	LAND_USE - : Total area [Units Square kilome...	348.0	0	0.0	0.0	0.0	0.0	0.0	0.0	
1	PAT_DEV - : Two and greater + All technologi...	193.0	0	0.0	0.0	0.0	0.0	0.0	0.0	
2	LAND_USE - : Land area [Units Square kilomet...	178.0	0	0.0	0.0	0.0	0.0	0.0	0.0	
3	PAT_DEV - : One and greater (all inventions)...	167.0	0	0.0	0.0	0.0	0.0	0.0	0.0	
4	YEAR	89.0	0	0.0	NaN	NaN	NaN	NaN	0.0	
	PAT_DEV - : Four and									

5	greater + All technolog...	39.0	0	0.0	0.0	0.0	0.0	0.0	0.0
6	PAT_DEV - : Three and greater + All technolo...	26.0	0	0.0	0.0	0.0	0.0	0.0	0.0

Wir sehen, dass das Feature das am häufigsten im Baum vorkommt "Land_USE: Total Area" ist, es geht also um die Fläche des Landes. Ein Blick in die Rohdaten des Features:

In [177]:
```
input_feature = pd.read_csv(Path(path_input / "aggregated-years") / "MATH-100.csv", index_col="Index")
relevant_feature = features.iloc[0,0]
input_feature = input_feature.loc[:, ["COUNTRY", "PISA-SCORE", relevant_feature]]
input_feature
```

Out[177]:

Index	COUNTRY	PISA-SCORE	LAND_USE - : Total area [Units Square kilometres]
AUS-2003	AUS	524.0	7741220.0
AUS-2006	AUS	520.0	7741220.0
AUS-2009	AUS	514.0	7741220.0
AUS-2012	AUS	504.0	7741220.0
AUS-2015	AUS	494.0	7741220.0
...	...	...	...
USA-2006	USA	474.0	9632030.0
USA-2009	USA	487.0	9831510.0
USA-2012	USA	481.0	9831510.0
USA-2015	USA	470.0	9831510.0
USA-2018	USA	478.0	9831510.0

231 rows × 3 columns

Wir stellen die Hypothese auf, dass das Feature als "Proxy" für das Land einsteht. Nun stellt sich die Frage, ob die Pisa-Daten innerhalb eines Landes stark streuen

In [211]:
```
from Global_configuartions import FOLDERNAME_PISA
import statistics

Pisa_Type = ["MATH", "READ", "SCIENCE"]
path_pisa = Path(FOLDERNAME_PISA)
country_csv = Path(FOLDERNAME_PISA) / "PISA_Länder.csv"
with open(country_csv, "r", encoding="utf-8-sig") as file:
```

```
    pisa_countries = file.read().splitlines()
country_overview_spread = pd.DataFrame(index=Pisa_Type, columns=pisa_count
ries)
country_overview_mean = pd.DataFrame(index=Pisa_Type, columns=pisa_countri
es)
for pisa_type in Pisa_Type:
    pisa_data = pd.read_csv(path_pisa/f"20-11-05_PISA_{str.lower(pisa_type
)}_total.csv", index_col="Index")
    countries = [x for x in pisa_countries if x in pisa_data["COUNTRY"].to
_list()]
    for country in countries:
        country_data = pisa_data.loc[pisa_data["COUNTRY"]==country, "value
"].to_list()
        if len(country_data) >= 2:
            country_overview_spread.loc[pisa_type, country] = statistics.s
tdev(country_data)
            country_overview_mean.loc[pisa_type, country] = statistics.mea
n(country_data)

for pisa_type in Pisa_Type:
    country_overview_spread_values = country_overview_spread.loc[pisa_type
].dropna().to_list()
    country_overview_mean_values = country_overview_mean.loc[pisa_type].dr
opna().to_list()
    country_overview_spread.loc[pisa_type, "Overall_Mean"] = statistics.me
an(country_overview_spread_values)
    country_overview_mean.loc[pisa_type, "Overall_Mean"] = statistics.mean
(country_overview_mean_values)
```

In [214]:
```
country_overview_spread["Overall_Mean"]
```

Out[214]:
```
MATH       8.392364
READ       9.770410
SCIENCE    7.871238
Name: Overall_Mean, dtype: float64
```

In [213]:
```
country_overview_mean["Overall_Mean"]
```

Out[213]:
```
MATH       484.668376
READ       484.673504
SCIENCE    489.160256
Name: Overall_Mean, dtype: float64
```

Man sieht, dass die durschnittliche Streuung des Pisa-Scores innhalb der einzelnen Länder mit ca. 2% des mittleren Pisa-Scores sehr gering ist. Somit kann man sagen, dass das Land ein guter Schätzer für den Pisa-Score zu sein scheint.

Für das betrachtete Feature (hier verwenden wir den Variationkoeffizienten als Parameter, da die Werte des Features der Länder stark auseinander gehen und ein Durschschnitt trügerisch sein kann)

In [216]:
```
countries_math = input_feature["COUNTRY"].unique()
feature_country_spread = pd.Series(index=countries_math)

for country in countries:
    country_data = input_feature.loc[input_feature["COUNTRY"]==country, re
```

```
levant_feature].to_list()
    if len(country_data) >= 2:
        feature_country_spread[country] = statistics.stdev(country_data)/s
tatistics.mean(country_data)

feature_country_spread = feature_country_spread.dropna()
feature_country_spread
```

```
<ipython-input-216-8e4bc392b6b0>:2: DeprecationWarning: The default dtype
for empty Series will be 'object' instead of 'float64' in a future version
. Specify a dtype explicitly to silence this warning.
  feature_country_spread = pd.Series(index=countries_math)
```

Out[216]:
```
AUS    0.000000e+00
AUT    0.000000e+00
BEL    0.000000e+00
BRA    0.000000e+00
CAN    0.000000e+00
CHE    4.877561e-06
CHL    3.571952e-04
COL    6.193187e-07
CZE    0.000000e+00
DEU    5.911524e-04
DNK    2.165163e-03
ESP    6.182408e-04
EST    1.330774e-03
FIN    3.461570e-04
FRA    2.230308e-04
GBR    0.000000e+00
GRC    0.000000e+00
HUN    0.000000e+00
IDN    1.279506e-03
IRL    0.000000e+00
ISL    0.000000e+00
ISR    0.000000e+00
ITA    9.885876e-04
JPN    7.844801e-05
KOR    3.221271e-03
LUX    0.000000e+00
LVA    5.349248e-04
MEX    1.039128e-06
NLD    9.828230e-05
NOR    2.664491e-01
NZL    0.000000e+00
POL    1.651504e-05
PRT    6.446614e-04
RUS    3.203385e-07
SVK    8.811460e-05
SVN    8.924092e-03
SWE    3.506176e-03
TUR    0.000000e+00
USA    1.054899e-02
dtype: float64
```

In [217]:
```
overall_average_spread = statistics.mean(feature_country_spread.to_list())
overall_average_spread
```

Out[217]: 0.007744030743096407

Man sieht, dass die Streuung innerhalb der Länder, wie angenommen, praktisch nicht existent ist. Nun stellt sich die Frage, wie gut das Feature allgemein mit dem Pisa Score korreliert.

In [218]:
```
# calculate correlation coefficient
input_feature["PISA-SCORE"].corr(input_feature[relevant_feature])
```

Out[218]: -0.18019516934420815

Da dieses Feature primär für die Güte des Modells verantwortlich ist, es nicht mit dem Pisa-Score korreliert, sondern lediglich eben wie die Pisa-Daten selbst marginale Schwankungen innerhalb der Länder aufweist, treffen wir nun die fundierte Annahme, dass das Feature räprentativ für das Land auftritt. Im Umkehrschluss gehen wir allerdings nun auch davon aus, dass die Größen des Features selbst für unsere modellierung irrelevant sind.

Nun stellt sich die Frage, ob dieses Phänomen auch bei anderen Features auftritt (wovon wir ausgehen). Dementsprechend führen wir ebendiese Betrachtung für alle Features aus. Das entsprechende Skript heißt: "Feature_Analysis.py" Die Ergebnisse laden wir hier rein:

In [5]:
```
feature_analysis = pd.read_csv(path_results / "Feature_Analysis_Country_Proxy.csv", index_col="Unnamed: 0")
feature_analysis
```

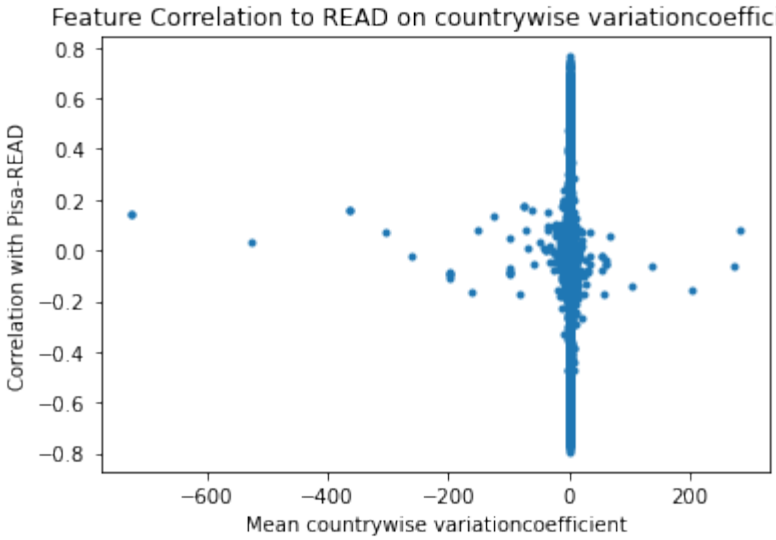
Out[5]:

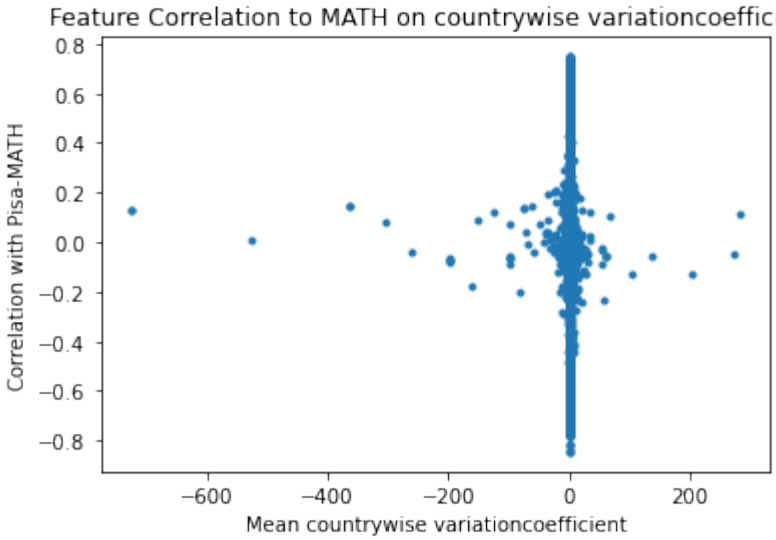
	Mean countrywise variationcoefficient	Mean countrywise st.dev	Mean countrywise average	R for READ	R for MATH	R f SCIENC
TABLE_I7 - : Average wage in US dollars based on Purchasing Power Parities [Units US Dollar]_x	0.220703	7417.961334	3.543917e+04	0.279572	0.291858	0.2389
TABLE_I7 - : Average wage in national currency units [Units National currency]_x	0.217087	435815.811982	1.584751e+06	0.141544	0.140489	0.0898
TABLE_I7 - : Personal income tax & employee social security contributions (All-in rate) [Units Percentage]_x	0.103251	4.065800	4.545714e+01	0.140993	0.110989	0.0960
TABLE_I7 - : Personal income tax [Units Percentage]_x	0.140216	4.335610	3.996398e+01	0.162382	0.099802	0.0870
TABLE_I7 - : Threshold (expressed as a multiple of the average wage) [Units Ratio]_x	0.467718	2.032746	4.008811e+00	-0.254846	-0.268354	-0.2415

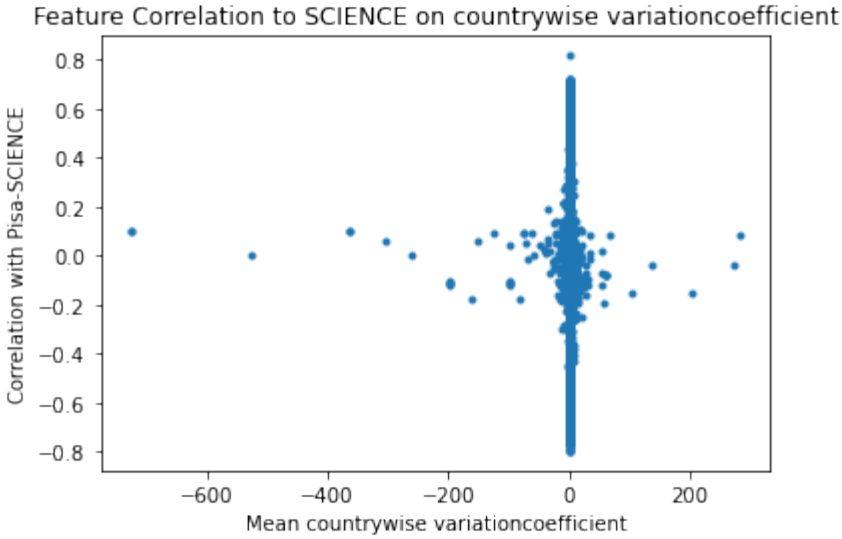
...	...	...	...	...	...	
SL.EMP.VULN.ZS-Vulnerable employment, total (% of total employment) (modeled ILO estimate)	0.068423	0.675133	1.919488e+01	-0.689025	-0.728907	-0.7211
SL.EMP.WORK.FE.ZS-Wage and salaried workers, female (% of female employment) (modeled ILO estimate)	0.009467	0.778385	8.062995e+01	0.674681	0.707787	0.7071
SL.EMP.WORK.MA.ZS-Wage and salaried workers, male (% of male employment) (modeled ILO estimate)	0.010366	0.741220	7.445580e+01	0.611657	0.668620	0.6581
SL.EMP.WORK.ZS-Wage and salaried workers, total (% of total employment) (modeled ILO estimate)	0.007931	0.646596	7.732000e+01	0.655825	0.700621	0.6950
SG.LAW.INDX-Women Business and the Law Index Score (scale 1-100)	0.035555	3.173789	8.896750e+01	0.522805	0.504804	0.4766

102734 rows × 6 columns

In [12]:
```
Pisa_Type = ["READ", "MATH", "SCIENCE"]
for pisa_type in Pisa_Type:
    plt.figure()
    plt.scatter(feature_analysis["Mean countrywise variationcoefficient"],
 feature_analysis[f"R for {pisa_type}"], marker='.')
    plt.xlabel("Mean countrywise variationcoefficient")
    plt.ylabel(f"Correlation with Pisa-{pisa_type}")
    plt.title(f"Feature Correlation to {pisa_type} on countrywise variatio
ncoefficient")
    plt.show()
    plt.close()
```

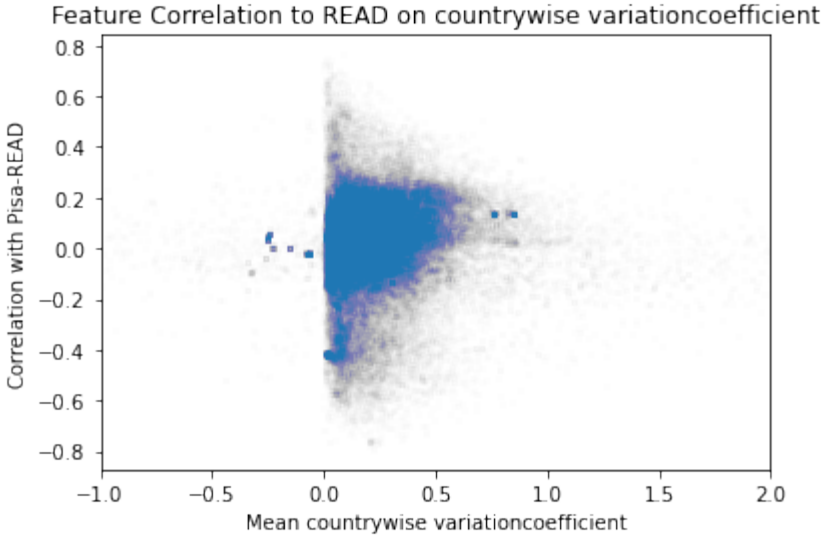


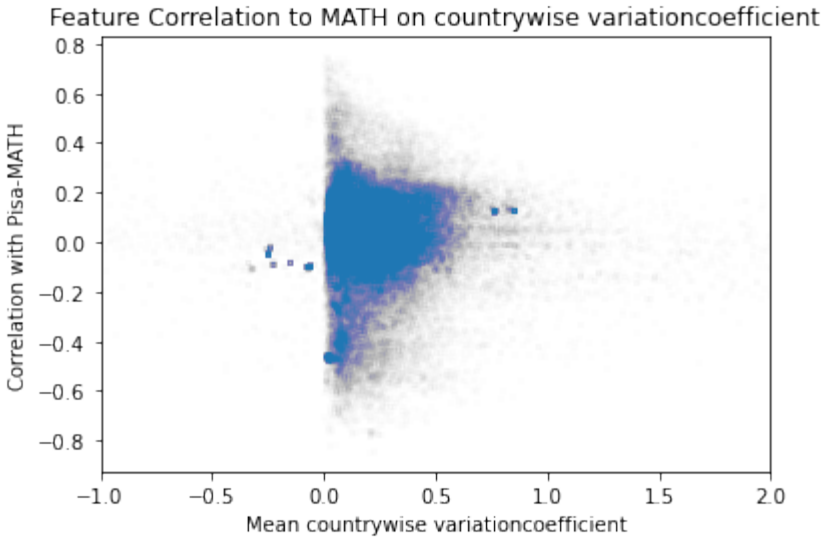


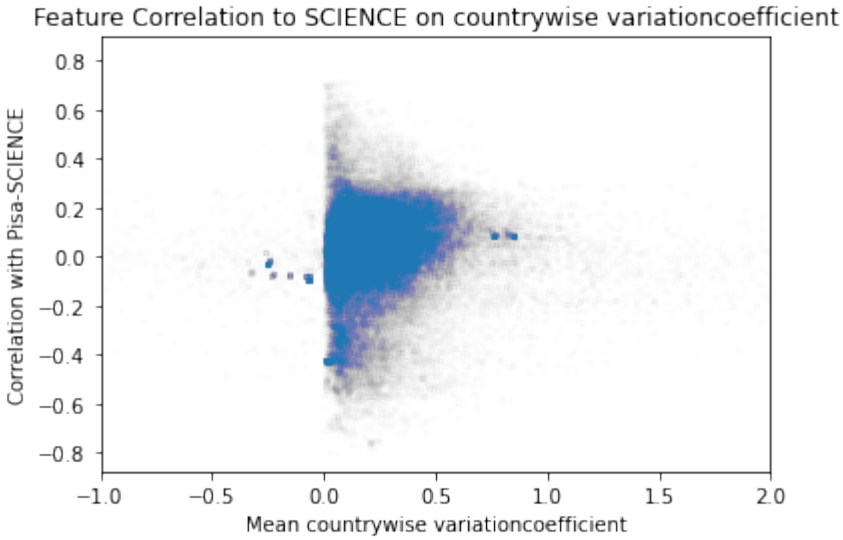


In [15]:
```
Pisa_Type = ["READ", "MATH", "SCIENCE"]
for pisa_type in Pisa_Type:
    plt.figure()
    plt.scatter(feature_analysis["Mean countrywise variationcoefficient"],
 feature_analysis[f"R for {pisa_type}"], marker='.', alpha=0.005)
```

```
    plt.xlabel("Mean countrywise variationcoefficient")
    plt.ylabel(f"Correlation with Pisa-{pisa_type}")
    plt.xlim(-1, 2)
    plt.title(f"Feature Correlation to {pisa_type} on countrywise variatio
ncoefficient")
    plt.show()
    plt.close()
```







Man sieht, dass ein sehr großer Anteil der Features in dem Bereich liegt, dass die Korrelation zu Pisa gering ist aber die Streuung innerhalb eines Landes niedlich ist. Wir stellen darauf basierend die Hypothese auf, dass der Großteil an Features im Baum dazu genutzt wird, die Werte des Features zu gruppieren und ihnen einen Pisa-Wert zuzuschreiben, nur weil die Pisa-Scores in der gleichen größenordnung vermutlich zum gleichen Land gehören.

Dieser Annahme zugrundelegend stellen wir fest, dass unser Entscheidungsbaum nicht dazu verwendet wird den Pisa-Score zu schätzen, sondern das Land zu schätzen.